ISBN 978-93-5156-328-0
International Conference of Advance Research and Innovation (ICARI-2015)

Database Security Measurements Issues in Adhoc Network

Reena Gangwar [a], M. K. Sharma [b]
[a] Department of Computer Science, Mewar University, Chittorhgarh, Rajasthan, India
[b] Department of Computer Science, Amrapali Institue , Haldwani, India

Article Info

Article history:
Received 4 January 2015
Received in revised form
10 January 2015
Accepted 20 January 2015
Available online 31 January 2015

Keywords

Database Security,
Security Level,
Control,
Attack,
Impact.

Abstract

Now-a-days database security at various level of internet applications and mobile applications related to social networking domain like Facebook and this has become key issue to protect recorded data from threats, unauthorized use, data loss, hackers, deletion, replication, unauthorized modification and false server. As in future when we will develop algorithms for mining the facts from these existing databases on web servers The complex nature of theses databases may create the security measures more complex to manage and control. Data is more insecure and vulnerable on network and web based distribution. The Probability of loss of data becomes increase if security measurement is not used at various level of warehouse designing and use of data. The paper discusses the related issues of security in web mining from data warehouse created by several web oriented database on various levels , to improve effectiveness of mining algorithms .

1. Introduction

Database system manages large bodies of information, which is vital to organizations. Any unauthorized access or manipulation of the database, therefore, spells trouble for the organization. Thus, database should be protected from the persons who are not authorized to access either certain parts of the database or the whole database. In other words, protecting database against unauthorized access or manipulation is a major concern.

There are some main control measurements to secure the data in databases.

2. Access Control

Restricting unauthorized access to the database is the main issue in developing secure DBMS. As discussed earlier, most database users need only a small portion of database to perform their job. Allowing them access to the whole database is undesirable. Thus, an organization should develop effective security policy to enable a group of users to access only a required portion of the database. Security policies of the organizations depend on the type of data maintained by them, hence, they vary from organization to organization. Further, once the security policy is developed, it should be enforced to achieve the level of security required.

- Access control is not a standalone component of a security system
- Access control coexists with other security services
- Access control works closely with audit control
- Access matrix is a good tool to specify permissions
- Access Control List (ACL) details are placed in Access Matrix

Two main approaches in DBMS for access control are discretionary access control and mandatory access control.

1.1 Discretionary Access Control

DAC is enforced in a database system by granting and revoking privileges to/from the users. Different users have different access privileges on the object (either a base table or a view) of the database. GRANT and REVOKE commands of data manipulation language corresponds to grant and revoke privileges, respectively. Discretionary Access Control (DAC) allows authorized users to change the access control attributes of objects, thereby specifying whether other users have access to the object. A simple form of Discretionary Access Control (DAC) might be file passwords, where access to a file requires the knowledge of a password created by the file owner. In Linux, the file permission is the general form of Discretionary Access Control (DAC).

Discretionary Access Control (DAC) is the setting of permissions on files, folders, and shared resources. The owner of the object (normally the user who created the object) in most operating system (OS) environments applies discretionary access controls. This ownership may be transferred or controlled by root/administrator accounts. Discretionary Access Control (DAC) is controlled by the owner or root/administrator of the Operating System, rather than being hard coded into the system.

The Discretionary Access Control (DAC) mechanisms have a basic weakness, and that is they fail to recognize a fundamental difference between human users and computer programs

1.2 Mandatory Access Control

Mandatory access control is a system-enforced method of restricting access to objects based on the sensitivity of the object and the clearance of the user. By contrast, Discretionary Access Control is enforced by individual file owners rather than by the system.

2. Inference Control

Even in multilevel secure DBMSs, it is possible for users to draw inferences from the Information they obtain from the database. The inference could be derived purely from the data obtained from the database system, or it could additionally depend on some prior knowledge which was obtained by users from outside the database system. An

Corresponding Author,
E-mail address: Reenagangwar108@gmail.com

inference presents a security breach if more highly classified information can be inferred from less classified information. There is a significant difference between the inference and covert channel problems. Inference is a unilateral activity, in which an unclassified user legitimately accesses Unclassified information from which that user is able to deduce Secret information. Covert channels, on the other hand, require cooperation of a Secret Trojan Horse which transmits information to an unclassified user by indirect means of communication. The inference problem will exist even in an ideal system which is completely free of covert channels.

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 inrelational systems is to augment a relational DBMS with a logic based inference engine and a knowledge base. The inference engine will detect security violations via inference when processing queries [30]. 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

3. Flow Control

Flow control regulates the distribution or flow of information among accessible objects. A flow between object A and object B occurs when a program reads values from A and writes values into B. Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects. Thus, a user cannot get indirectly in B what he or she cannot get directly in A. Active flow control began in the early 1970s. Most flow controls employ some concept of security class; the transfer of information from a sender to receiver is allowed only if the receiver's security class is at least as privileged as the sender's.

A flow policy specifies the channels along which information is allowed to move. The simplest flow policy specifies just two classes of information: confidential (C) and non-confidential (N), and allows all flows except those from class C to class N. This policy can solve the confinement problem that arises when a service program handles data such as customer information, some of which may be confidential. For example, an income-tax computing service might be allowed to retain a customer's address and the bill for services rendered, but not the customer's income or deductions.

Access mechanisms are responsible for checking user's authorizations for resource access: only granted operations are executed. Flow controls can be enforced an extended access control mechanism, which involves assigning a security class (usually called a clearance) to each running program). The program is allowed to read a particular memory segment only if its class is as low as that of the segment. This automatically ensures that no information transmitted by the person can move from a higher to a lower class. For example, a military program with a secret clearance can only read from objects that are unclassified and confidential and can only write into objects that are secret or top secret.

Covert channels can be classified into timing (temporal) and storage (spatial) channels. One way to think of the difference between covert timing channels and covert storage channels is that covert timing channels are essentially memory less, whereas covert storage channels are not. With a timing channel, the information transmitted from the sender must be sensed by the receiver immediately, or it will be lost. However, an error code indicating a full disk which is exploited to create a storage channel may stay constant for an indefinite amount of time, so a receiving process is not as constrained by time.

4. Data Encryption

We describe the different database encryption options-showing how each variant affects application performance, manageability, and security. We'll cover both traditional encryption and the newer alternatives: tokenization and Format Preserving Encryption. We also review key management options, the role of access controls, and application-level encryption. Finally, we close with a real-world example of how to select a database encryption or tokenization solution to meet your organization's security objectives. We feel the best way to accomplish these goals is to present a simple guide for selecting a database encryption strategy - basically a decision tree for you to plug in your requirements and map those to the available database security choices. This process will highlight the differences in approaches, and tie business needs to technical capabilities. Between this guide and the use cases, end users will have the tools to both clarify their goals and determine an appropriate implementation strategy.

The term database encryption is used to describe many different methods of data protection, implemented either outside or within the database engine. Conceptually, a database is a sophisticated box to put data in. Taking this analogy one step further, you can protect the entire box (File/OS), the entire contents of the box (Full database), or some subset of the content within the box (Column, Table, Schema). We can apply encryption to the contents through native database functions or externally with third party tools, but both are called database encryption. We also see growing use of *tokenization* as an alternative or complement to encryption, since it achieves many of the same goals, For this discussion, we divide database encryption into two basic types:

4.1 Transparent/External Data Encryption (TDE)

These terms refer to encryption of the entire database. This is provided by native encryption functions within the database engine. Some database vendors offer column and table level granularity, but it is increasingly common to apply encryption to all the data. We call this 'transparent' database encryption because it is invisible to the applications and users that use the data, and requires no changes to application logic. The principal use case is to prevent exposure of information due to loss of the physical media (disk, tape, etc.) or compromise of the database files in storage.

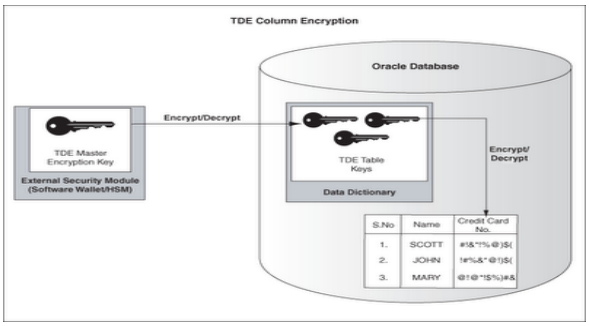

Fig: 1. Column level TDE

Transparent encryption can also be handled through drive or OS/file system encryption, applying encryption on everything that gets written to disk. Although these options may lack some of the protections of native database encryption, both are invisible to the application and do not require alterations to the code or schemas. Transparent encryption protects the database from users without database credentials, but does not protect data from authorized users.

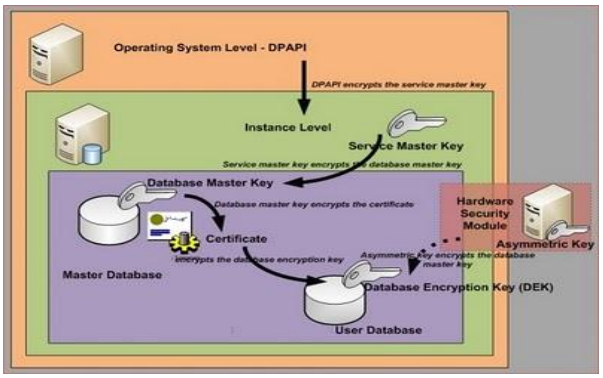

Fig: 2. Transparent Encryption

4.2 User/Data Encryption

These terms describe encrypting specific columns, tables, or even data elements within the database. We call this 'user' encryption because the objects being encrypted are owned and managed on a per-user basis. Tokenization also falls into this category. The classic use case for this encryption model is encrypting credit card numbers within a database. The goal is to provide protection against inadvertent disclosure, or to enforce separation of duties on credentialed users of the database. The downside is that these variants are not invisible to the application and usually require code and database changes. The concept is to encrypt only the highly sensitive data we are worried about, reducing the overall performance impact, and minimizing code and database changes. How this is accomplished depends on how key management is handled, the use of internal vs. external encryption services, and how applications use the database.

It should be stressed that *both* user and transparent encryption protect media, but the granularity provided by user encryption comes at the cost of required modifications to code and/or database schemas. Some vendors offer transparent encryption applied to specific tables or columns, but the value proposition is still focused on lost media and file protection, not separation of duties.

Tokenization or Format Preserving Encryption are designed to reduce required external application changes, but still require internal database modifications. In some of our earlier essays and articles, we described these two options as "Encryption for Media Protection" and "Encryption for Separation of Duties". That was a bit of an oversimplification, but those terms better describe the value they provide vs. the technology features that do the work. While we wanted to use terminology that directly mapped to the business use case, if you asked your RDBMS vendor for "media encryption", they would be unlikely to know what the heck you were talking about.

In summary, transparent/external encryption protects data from compromise by attacks from outside the database, but does not protect against credentialed database users. User/data encryption can encrypt data and restrict access based on database users, but at a higher cost in terms of complexity, performance, and manageability.

The security of data is the most important, Database security covers the following issues.

- **Privacy**: Only authorized persons should be allowed to access the database. In addition, only the part of the database that is required for the functions they perform should be available to them. In other words, users are allowed to access only the information that is pertinent to their jobs.
- **Database integrity**: Database should be protected from improper modifications, either intentional or accidental, to maintain database integrity. Only the type of operations that need to be performed by the user should be allowed to them. For example, an employee who does not belongs to accounts department should not be allowed to modify the balance sheet of the organization. The employees of accounts department only should be allowed to do so.
- **Database availability**: Security should not restrict the authorized users to perform their actions on the part of the database available to them. For example, an accounts department employee should not be restricted to update the balance sheet.

5. Conclusion

In this paper, the security issues and requirements for discretionary and mandatory security models for the protection of conventional database systems and object oriented database systems are illustrated. We have also discussed the issues related to security in object-oriented databases. Several proposals for discretionary and mandatory security models for the protection of conventional databases and object-oriented database systems are presented. Still, there is not a standard for

designing these security models. The work presented in this paper gives a collected picture of different security issues of database; it can be extended to define, design and implement an effective security policy on a database security.

References

[1] E. Bertino, E. Ferrari, Administration Policies in a Multipolicy Authorization System, Proc. 10th Ann. IFIP Working Conf. Database Security, 1997

[2] E. Bertino, S. Jajodia, P. Samarati, An Extended Authorization Model, IEEE Trans. Knowledge and Data Eng., 9(1), 1997, 85-101

[3] M. Zand, V. Collins, D. Caviness, A Survey of Current Object-Oriented Databases, ACM SIGMIS Database, 26(1), 1995

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

[5] M. S. Olivier, S. H. V. Solms, A Taxonomy for Object-Oriented Secure Databases, ACM Transactions on Database Systems, 19(1), 1994, 3-46

[6] Fausto Rabitti, Elisa Bertino, Won Kim, Darrell Woelk, A Modelof Authorization for Next-Generation Database Systems, ACM Transactions on Database Systems (TODS), 16, 1991

[7] Pierangela Samarati, Elisa Bertino, Alessandro Ciampichetti, Sushil Jajodia, Information Flow Control in Object-Oriented Systems, IEEE Transactions on Knowledge and Data Engineering, 9(4), 1997, 524–538

[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, 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, 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, C. Wood, Database Security and Integrity, Addison-Wesley, 1981

[12] Elisa Bertino, Ravi S. Sandhu, Database Security - Concepts, Approaches, and Challenges, IEEE Transactions on Dependable and Secure Computing, 2(1), 2005, 2 –19

[13] J. M. Slack, E. A. Unger, A Model of Integrity forObject-Oriented Database Systems, Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing: technological challenges of the 1990's, 1992

[14] R. Fagin, On an Authorization Mechanism, ACM Trans. Database Systems, 3(3), , 1978, 310-319

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

[16] E. Bertino, P. Samarati, S. Jajodia, High Assurance Discretionary Access Control for Object Bases, Proceedings of the 1st ACM conference on Computer and communications security, 1993

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

[18] E. Bertino, P. Bonatti, E. Ferrari, TRBAC: A Temporal Role- Based Access Control, ACM Trans. Information and SystemSecurity, 4(3), 2001, 191-233

[19] E. Bertino, C. Bettini, P. 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. Bertin, L. M. Haas, Views and Security in Distributed Database Management Systems, Proc. Int'l Conf. Extending Database Technology, 1988

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

[22] J. Richardson, P. Schwarz, L.-F. Cabrera, CACL: Efficient Fine-Grained Protection for Objects, ACM SIGPLAN Notices, conference proceedings on Object-oriented programming Systems, languages, and applications, 27(10), 1992.

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

[24] Oracle, The Virtual Private Database in Oracle9iR2, available athttp://otn.oracle.com/deploy/security/Oracle9iR2/pdf/VPD9ir2wp.pdf,2000.

[25] HIPAA, Health Insurance Portability and Accountability Act of 1996, available at http://www.hep-calert.org/links/hippa.html,1996

[26] A. Baraani-Dastjerdi, J. Pieprzyk, R. Safavi-Naini, A Security Model for Multi-Level Object-Oriented DatabasesBased on Views, Tr-96-03, Department of Computer Science, The University of Wollongong, Wollongong, Australia, 1996

[27] D. E. Denning, T. F. Lunt, A Multilevel Relational Data Model, in Proceedings on Symposium on Computer Security and Privacy, IEEE Computer S ociety Press, 220-234, Oakland, CA., 1987

[28] P. J. Denning, A Lattice Model of secure Information Flow, Comm. ACM, 19(5), 1976, 236-243

[29] M. B. Thuraisingham, Mandatory Security in Object-Oriented Database Systems, ACM SIGPLAN Notices, Conference proceedings on object-oriented programming systems, languagesand applications, 24(10), 1989

[30] T. F. Keefe, SODA: A Security Model for Object Oriented Database Management Systems, Proceedings of the I5fhnternational Computer Software and Applications Conference, Tokyo, Japan, September 1991

[31] C. Zaniolo, Object-Oriented Programming Prolog, Proceedings of the IEEE Logic Programming Symposium, 1984