International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Database Security Issues: A Review

Ogbonna J. C.[1], Nwokoma F. O.[2], Ejem A.[3]

[1, 2, 3]Department of Computer Science, School of Physical Sciences, Federal University of Technology Owerri, Imo State Nigeria.

Abstract: ***Database security such as confidentiality, integrity, and availability of data/information stored in a database is one of the most crucial and major challenges in the world of Information and Communication Technology. Inaccuracy/Loss of data stored in a database in some cases may be regarded as a loss of human life. That is to say that, database may provide false information that could render the entire organization hopeless if not well protected, and so techniques that protects database from attackers, and any kind of corruption should be carefully implemented and should be given a higher priority. The major factors to consider in database security issues include how to prevent unauthorized access to data, how to prevent unauthorized manipulation of data, as well as how to insure the availability of data when needed. This paper aims at reviewing various database security issues such as data confidentiality requirements, data integrity requirements, and data availability requirements.***

Keywords: Database security, security techniques**,** database integrity, database threats

1. Introduction

Database security is refers to protecting the database system from corruption, unauthorized or malicious use. It is concerned with limiting the users to performing only those operations they are allowed, and to ensure proper execution of transactions despite failures. It is also one of the most challenges associated with database systems in today's information system infrastructure. The primary objectives of database security are to prevent unauthorized access to data, prevent unauthorized tampering or modification of data, and to insure that data remains available when needed [1]. As a result of adoption of database system for business transactions and decision making systems, security related issues has to be considered in order to protect valuable data from unauthorized data disclosure (*loss of confidentiality*), incorrect data modification (*loss of integrity*), and data unavailability (*loss of availability*). This paper will review various database security issues such as data confidentiality requirements, data integrity requirements, and data availability requirements.

2. Database Security Requirements

Database security is built upon a framework encompassing three constructs [2], as illustrated in Figure 1: database confidentiality or secrecy (authentication and authorization), database integrity (which has to do with the accuracy, validity, correctness and consistency of data stored in a database system), and database availability (which has to do with ensuring that data remains available when needed).

The next sections focuses on the confidentiality requirement and discuses authentication methods, and several access control mechanisms currently used to address issues relating to authorization requirement. Finally, we discuss how to maintain database integrity as well as the strategies and procedures involved in protecting the database against data loss, and reconstructing the database after any kind of failure (data availability).

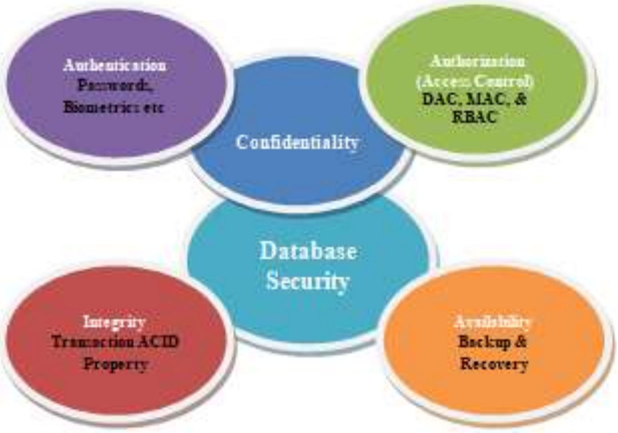

Figure 1: Database Security Techniques

3. Database Authentication Methods

User authentication/identification is normally required before a user can access the database. Authentication means verifying the identity of someone (a user, device, or an entity) who wants to access data, resources, or applications. Validating that identity establishes a trust relationship for further interactions [3]. To validate the identity of database users, we employ the combination of OS authentication, remote database authentication, local database authentication etc.

3.1 Operating System Authentication

Once authenticated by the OS, users can connect to the database without specifying their usernames and passwords.

3.2 Remote Database Authentication

Authentication over network is usually handled by remote user authentication services such as Secure Socket Layer (SSL) protocol, Password Authentication Protocol (PAP), Shiva PAP (SPAP), Challenge Handshake Authentication Protocol (CHAP), Microsoft CHAP (MS-CHAP), Extensible Authentication Protocol (EAP), Remote Authentication Dial-In User Service (RADIUS) etc.

3.3 Local Database Authentication

Database Administrator (DBA) creates/manages usernames and passwords for all the users accessing the database. The users must provide their username and password when attempting to establish a connection. This process prevents unauthorized use of the database by denying connection to the database if a user provides incorrect credentials. Database Authentication includes the following:

a) **Password Encryption:** Database should always perform an automatic password encryption during network connection before sending them across the network.
b) **Password Complexity Verification:** A verification check is carried on a password whether it is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords. It is recommended to use a combination of uppercase and lowercase letters, numbers, special characters, and even to specify a reasonable minimum length for password characters.
c) **Account Locking:** Database should handle brute force attack [4], by locking user's account for a specified duration of time, after a specified number of successive failed login attempts.

Other authentication types other than password authentication include:

a) **Physical authentication** such as smart cards, digital certificates etc [3].
b) **Biometric authentication** such as signatures analysis devices, fingerprints, retina eye scans and voice analysis devices etc [4], [5].

4. Database Authorization Methods

Authorization methods protect data against unauthorized disclosure [1], [6]. It simply implies assigning only certain users the privilege to access, process or alter certain data, and this is mainly addressed by access control mechanisms and supported by the use of encryption techniques. A privilege is a right to execute a particular type of SQL statement or to access another user's object. For instance, a user could be given a privilege such as: (a) connect to a database, (b) create a table, (c) retrieve rows from a table crated by another user, (d) execute a stored procedure created by another user, (e) modify a table created by another user etc.

A privilege (access control) should be granted only to users who require such a privilege in order to accomplish their task. Excessive granting of unnecessary privileges can compromise database security [3].

4.1 Database Access Control Mechanisms

Access control is the process by which rights and privileges are assigned to users and database objects. The basic concepts in access control models are authorizations, subjects and objects. An authorization is a statement whether a given subject can perform a particular action on an object. The subject is a user and the object is the data item this user is trying to access [6]. Access Control Mechanisms can be divided into three categories: Discretionary Access Control (DAC), Role-Based Access Control (RBAC) and Mandatory Access Control (MAC).

4.1.1 Discretionary Access Control (DAC) Mechanisms

Discretionary access control mechanisms govern access to data based on the identity of the subject and authorization rules. DAC mechanisms allow subjects to grant other subjects access to data.

In Figure 2, a subject has an arbitrary number of permissions (authorizations) which relate operations (access modes) to objects. A permission relates one operation to one object but an operation or an object can be used in multiple permissions [6].

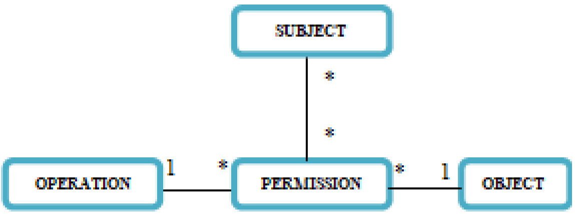

Figure 2: Discretionary Access Control Mechanisms

Database administrators or object owners grants or revokes authorization. Then the question is: what happens when a user that has delegated authorizations loses his or her authorizations? What happens to the authorizations this subject granted to other subjects? The solution to this is that each time an authorization is revoked, granted authorizations are revoked recursively. This approach can be very disruptive, and a different approach to this mechanism is the introduction of roles and the assignment of privileges to roles instead of assignment of privileges directly to users (subjects). This mechanism is known as Role-Based Access Control (RBAC) [2].

4.1.2 Mandatory Access Control (MAC) Mechanisms

Mandatory Access determines access to data based on classifications of subjects and objects. According to [6], the most widely used model for MAC is the Bell-LaPadula model. To illustrate this model, let's consider a security classification with the four security levels Top Secret (TS), Secret (S), Confidential (C) and Unclassified (U). These security classifications are given in the order TS > S > C > U. As illustrated in Figure 4, Bell-LaPadula model [7] enforced two restrictions as follows:

a) **No read-up:** A subject can only read objects if class (Subject) >= class(Object). Thus a subject with the classification Confidential (C) can only access objects classified as Confidential (C) and Unclassified (U). This is also known as the simple security property.
b) **No write-down:** A subject can only write objects if class(Subject) <= class(Object). Thus a subject with access classification Confidential (C) can only write objects with the access classification Confidential (C), Secret (S) and Top Secret (TS). This is also known as the star property.

In Figure 3, the subject having a security classification of Confidential (U) can read any object with a security classification less than or equal to Confidential and only write objects with a security classification greater than or equal to Confidential.

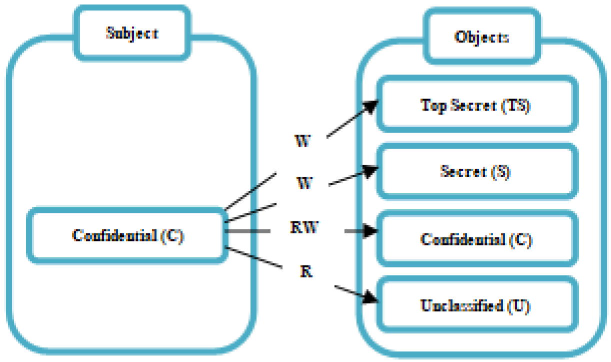

Figure 3: Mandatory Access Control Mechanisms

4.1.3 Role-Based Access Control (RBAC) Mechanisms

BAC is an access control mechanism used to simplify the process of granting and revoking authorizations (authorization administration). Here, authorization of an operation to a given object is not directly granted to subjects (users), rather they are granted to a role associated to the operation. Whenever a user (subject) needs to perform a particular task or operation, he only needs to be granted the permission to play the role associated with the task or operation. If the task or operation changes, only the right to play a given role needs to be revoked.

In Figure 4, a subject is assigned a role or multiple roles. Each role contains permissions that relate a particular operation to a particular object. Roles can contain other roles, which establish role hierarchies. Every role can contain several permissions, which relate one operation to one object. One object or operation can be used in several permissions

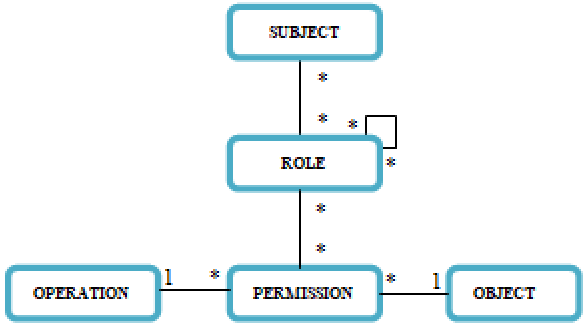

Figure 4: Role-based Access Control Mechanisms

Based on the fact that RBAC includes the concept of role hierarchies allowing role to sub-role relationship, there is need to avoid problems associated with excessive granting of unnecessary privileges which may compromise database security. To solve these problems, RBAC employs a technique called Separation of Duty (SoD) constraints, which is a technique that prevents a subject from receiving too many authorization [6].

4.2 Comparison of DAC, MAC and RBAC

a) Discretionary Access Control mechanisms offer a high degree of flexibility that makes them suitable in many application domains. However, they are vulnerable to malicious attacks because they do not impose any control on how information is propagated after authorized users access it [6].
b) Mandatory Access Control mechanisms prevent this illegal flow of information, but tend to be inflexible because they require a rigid classification of subjects and objects in security classes and enforce the 'No Read-Up' and 'No Write-Down' constraints and this restricts their applicability [6].
c) Role-based Access Control mechanism is a flexible alternative to both MAC and DAC. The main difference between DAC and RBAC is that authorization policies usually are not assigned to individual users but to groups of users, also known as roles. Each user is entitled to play one or more roles and exercise the associated authorizations [6].

5. Database Integrity

Database integrity has to do with the accuracy, validity, correctness and consistency of data stored in a database system. That is, integrity is concerned with making sure that operations performed by users accessing the database are correct and maintain database consistency even in an event of failures. A condition or restriction applied to given set of data used to minimize data entry error is commonly referred to as integrity control. In a relational database model, integrity controls may be applied to individual attributes, the relationship between two different attributes, or the relationship between tuples of one or more relations. Enforcement of integrity constraints will be carried out automatically by the DBMS as each data item is entered.

Another concept associated with database integrity is refers to as Database Transaction. A transaction is a logical unit of work that must be either entirely completed or aborted. No intermediate states are accepted. That is to say that a transaction is a program unit whose execution preserves the consistency of the database. If the database is in a consistent state before a transaction executes, then the database should also be in a consistent state after its execution. To ensure integrity of the data, database system must maintain the ACID properties of database transactions [8].

5.1 Database Transaction ACID Properties

a) **Atomicity:** Atomicity requires that all operations of a transaction be completed otherwise the transaction will be aborted entirely. That is, it represents the central issue that all part of the transaction must succeed or fail together.
b) **Consistency:** Consistency implies that all data in the database must ultimately be consistent even though there might be temporarily inconsistencies while a transaction is being processed, however, in the end, the database must be returned to a consistent state.

c) **Isolation:** Isolation means that concurrent access problems are prevented, i.e. the data used during the execution of a transaction cannot be used by a second transaction until the first transaction is completed. This is to ensure that changes made by one transaction do not result to an error by other transactions.
d) **Durability:** Durability implies that the consistent state of the database is always permanent, i.e. when a transaction is completed, the database reaches its consistent state and this state can never be lost even in an even of system failure.

5.2 Concurrency Control

Concurrency control is one of the integrity constraints employ to solve the concurrent access problems. It is the coordination of simultaneous execution of transactions on multi-processing database system. It ensures serialization of transactions in a multi-user database environment. The problems which concurrency control tries to solve include:

a) **Lost Update:** This occurs when transactions T_1 and T_2 are executing concurrently, and T_1 reads a data item, modifies it, and before T_1 could commit the change to the database T_2 has already accessed the same data item and modifies it as well. The update performed by T_1 will be lost if T_2 commits its changes after T_1 commits.
b) **Uncommitted Data:** This occurs when T_1 and T_2 are executing concurrently and T_1 is rolled back after T_2 has already accessed the uncommitted data thereby violating isolation property.
c) **Inconsistent Retrieval:** This occurs when a transaction is calculating summary or aggregate function when other transactions are updating the same data. E.g. T_1 may be calculating (a) the count of existing airtime (b) the count of used airtime and (c) the count of unused airtime, while T_2, T_3, …, T_n are loading airtimes. This could lead to an inconsistent retrieval such as (a) = 100, (b) = 60 and (c) = 30. I.e. 10 additional airtimes were loaded between the operations (b) and (c) of T_1 by the other Ts, thereby resulting in an inconsistent retrieval.

5.3 The Scheduler

Concurrency control uses a scheduler to establish the order in which operations with concurrent transactions are executed. To determine the appropriate order, the scheduler bases its action on concurrency control algorithms such as locking or time stamping methods, and tries to optimize CPU usage. If there is no better way to execute the transactions, the scheduler uses FCFS basis, which may not produce an optimum CPU usage.

5.4 Transaction Deadlock

Introduction of locking policy by the scheduler in order to ensure serialization of transactions brought about another problem called 'Deadlock' which arises when two or more transactions have placed a lock on data and are waiting for the other's data. To solve this problem, one of the transactions has to backtrack and release its lock. Two common solutions exist for deadlock problem:

a) The first approach is that transactions wait for random length of time, release all its existing locks, and start all over again. This will enable other transactions to proceed with their execution, make way for other transactions. The drawback of this method is that transaction spends a lot of time waiting.
b) The second approach is called two-phase locking. Here the locking operations are (a) READ_LOCK, which allows the reading of a data item and (b) WRITE_LOCK, which allow both reading and writing of a data item.

6. Database Availability

Availability refers to the prevention and recovery from hardware and software failures as well as from malicious data access resulting in the denial of data accessibility [2]. Information stored in a computer media is subject to loss on corruption caused by a wide range of events and is important to provide ways of restoring the database to precisely the state that existed at the time of system failure so that data will always be made available to users who need them.

6.1 Sources of Failure

Several types of failure that can halt the normal operation of a database include [8]:
a) **System failures** such as deadlock, power interruption, operating system failures etc.
b) **Hardware failures** such as disk failure, loss of transaction signal over the transmission link.
c) **Logical errors** such as bad data or missing data, data type errors, calculation error such as division by zero, square root of a negative number etc.

6.2 Recovery Procedures

Recovery refers to the various strategies and procedures involved in protecting the database against data loss and reconstructing the database after any kind of failure, and avoid or reduce the possibility of having to duplicate work manually. Recovery procedures vary depending on the type of failure that occurred, the structures affected, and the type of recovery performed [8]. Any recovery procedure must maintain database integrity by ensuring that any transaction must be in either ABORTED or COMMITTED state. A committed transaction should always leave the database in a new consistent state, and any incomplete transaction should be in aborted state. Aborted state tells the DBMS that any other transaction to be executed should wait until the database is restored to the state it was before the transaction that caused the failure stated, and a transaction log plays an important role in this case.

6.3 Log-based Recovery

The transaction log plays a key role in failure recovery. The log is the history of all the changes made to the database as well as the status of each transaction. It is obviously important that the data on the log should not be lost, damaged or destroyed. Whenever a transaction performs a write operation, it is essential that the log record for that write be created before the database is modified. For log records to be useful for recovery from system and disk failures, the log must reside in a stable storage [8]. In practice this means keeping multiple copies of transaction log on disks residing in different locations.

6.4 Database Recovery Features

There are four main database recovery features: mirroring, reprocessing, roll-forward, and roll-backward.

a) **Mirroring:** This implies frequent simultaneous copying of the database to two or more different locations that could be used to recover the database in an event of any failure. Actually, this is an expensive recovery technique.
b) **Reprocessing:** In reprocessing, a DBA goes back to the point before the failure and reprocess the workload from there. Here, periodic database copies must be made and all the transactions executed after the most recent copy/backup of the database should be kept in a file. When there is a failure, the information in the file can be re-entered by the users or DBA and reprocessed together with the most recent copy/backup of the database. This technique is so stressful and very time consuming.
c) **Roll-forward (Forward Recovery):** In roll-forward, periodic database copies are also made and all the transactions executed after the most recent copy/backup of the database are kept in a transaction log. When there is a failure, the information in the log together with the most recent copy/backup of the database automatically is used to recover the database to the state it was before the failure.
d) **Roll-backward (Backward Recovery):** Roll-backward is used to undo unwanted changes to the database since the last update.

In each of these recovery features, the database will be accessible (available) to other transactions only when the database is restored to the expected consistent state.

7. Conclusion

In this paper, we reviewed various database security issues such as data confidentiality, data integrity, and data availability. In addition, we presented some common security techniques that can be employed to enhance the security of the database against some known attacks and security threats. In section 1, we introduced the database security issues and the need to secure information in a database. The rest of the sections went in details to explain these security issues and the techniques that resolve these security issues.

References

[1] M. C. Murray, “Database Security: What Students Need to Know,” *J. Inf. Technol. Educ.*, vol. 9, pp. 61–77, 2010.
[2] E. Bertino and R. Sandhu, “Database security-concepts, approaches, and challenges,” *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 1, pp. 2–18, 2005.
[3] D. Gosselin and R. Smith, “Oracle ® Database 10g, Security Guide,” vol. 2, no. July, 2012.
[4] B. Abdulrahman Hamed Almutairi, A. Helal Alruwaili, A. Hamed Almutairi α, and A. Helal Alruwaili α, “Security in Database Systems Security in Database Systems Security in Database Systems,” *Glob. J. Comput. Sci. Technol. Netw.*, vol. 12, no. 17, 2012.
[5] M. R. Shinde, “Overview of Database Security,” vol. 5, no. 6, pp. 7920–7921, 2014.
[6] J. Van Loon, “Database Security Concepts and Approaches,” 2017.
[7] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems, Fourth Edition*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.
[8] P. R. Patel, “Database Recovery Techniques : A Review,” vol. 4, no. 4, pp. 11482–11486, 2015.

Author Profile

Ogbonna James C. obtained his B.Tech. and M.Sc. degrees in Computer Science from Federal University of Technology Owerri in 2009 and 2016, respectively. He is a Ph.D student at the Federal University of Technology Owerri. His research interest includes Mobile Computing, Information Security, Database Management Systems, Object-oriented Design, Software Engineering, and Programming.

Nwokoma Francisca O. obtained a National Diploma in Computer Science from Abia State Polytechnic Aba in 2005; B.Tech. and M.Sc. in Computer Science from Federal University of Technology Owerri in 2010 and 2016 respectively. She is an academic Staff of Computer Science Depts., FUTO. Her research interest includes Data Communication and Networking, Software Engineering, and Human Computer Interaction.

Ejem Agbaeze obtained a National Diploma in Computer Science from Abia State Polytechnic Aba in 2003; B.Tech. and M.Sc. in Computer Science from Federal University of Technology Owerri in 2008 and 2015 respectively. He is an academic Staff of Computer Science Depts., FUTO. His research interest includes System Modelling and Simulation, Cloud Computing and Distributed Systems, Optimization Algorithms.