A STUDY ON DATABASE SECURITY ISSUES AND OVERCOME TECHNIQUES

Dr.P.Chandra Kanth[1], Y.Vengalarao[2], S.V.Aravind Kumar[3], P.L.Chaithanya[4], S.Himaja[5]

Associate Professor[1] ,Department of CSE[1],
UG student[1],UG student[2], UG student[3], UG student[4],
Department of Computer Science and Engineering,
Audisankara College of Engineering and Technology, Gudur, India

---

Abstract : Nowadays Databasesecurity has become an important issue in technical world. The main objective of database security is to forbid unnecessary information exposure and modification data while ensuring the availability of the needed services. A number of security methods have been created for protecting the databases. Many security models have been developed based on different security aspects of the database. All of these security methods are useful only when the database management system is designed and developed for protecting the database. Recently the growth of web applications with database at its backend Secure Database Management System is more essential than only a Secure Database. Therefore this paper highlights the Threats, Security Methods and Vulnerabilities in Database Management System with the help of surveys performed in the field of secure databases.Propose approach for database security that exploits existing DBMS facilities to associate a separately maintained checksum value with critical data. Using our approach, a database's content, domain and referential integrity remain the responsibility of the DBMS, however, when critical data is manipulated checksum values are computed and stored in a separate database.

IndexTerms **- Component,formatting,style,styling,insert.**

I. INTRODUCTION

These days, including the invention of internet technology, securing database is a needed aspect in today's world. Individually we use database every day unknowingly when we browse the internet. The information we get on the web page is the consequences of query accomplished by the webpage to the database it is connected. Hence indirectly via the webpage we are connected to different databases. The web pages are open for any anonymous person in the world or we can say the databases are indirectly opened for everyone. As we know data in the database is the most valuable asset which can be the source of information. All the information cannot be revealed for everyone. Hence many security tools have been devised to protect the database. As the database is accessible via web pages security should be implemented in database management system (DBMS).Looking towards the implementation this paper focus on Vulnerabilities in Database Management System (VDBMS), Threats in Database Management System (TDBMS) and Security Methods in Database Management System (SMDBMS). Rest of the paper is organized as follows: section II provides overview of recent trends in database protection, section III, IV and V are devoted to VDBMS, TDBMS and SMDBMS. Section VI deals with the summary of above section in a tabular format and section VII deals with the conclusion.

The primary responsibility for ensuring the security and integrity of a database lies with the **D**ata**b**ase **M**anagement **S**ystem (DBMS) . At present, the most widely used **R**elational DBMS's (RDBMS's) provideseveral ways of ensuring data integrity at the entity, domain, referential, and user-defined levels. The latter of these levels, i.e. the user defined level, involves constraints on the forms of update that can be performed, and such constraints are usually enforced via triggers either by the database itself or by applications that access the database.
Checksums have been the subject of research and practical application for many years. Gopalan used checksums to enhance the integrity of a conventional file system via ablock-level checksum computed per-block for all data blocks in a file, which is indexed by arelative block number. In such an approach, files are modified to include a set of additional references that point to the checksum blocks, and the addition of a block to a file involves the computation and storage of its checksum blocks. Crocker introduced a checksumbased "Spam Detection" engine for Lotus Domino mail systems called *Block* which also exploitschecksums[1-5]. The system computes a checksum for each message that is classified as "Spam" and maintains the checksums in a database which is replicated between the mail system and the mail server. The database enables the server, on receipt of each message, to compute a checksum and compare it with those checksums already in the database. If a match is found, it is likely that the associated message is further "Spam". NetworkAppliance Inc. technical report describes a technique for reducing the volume of data transfer during Backup and Restore on UNIX platforms. The system uses a checksum to identify portions of a file that have changed since a previous and current backup. Changedblocks are identified via checksum values computed and maintained for each block. Sabartnam , uses checksums to detect errors in database manipulations including transactions, locks, logs, and data buffers. In this case, a checksum is added to the object being manipulated, and a checksum field

is attached to the access method and also to each of the objects in associated hash buckets. A hash vector and object checksum are recomputed during the restart of a DBMS and compared with stored checksum values. We describe an approach to database security that similarly exploits checksums. First the forms of data integrity that must be maintained in a conventional RDBMS are examined. We present our approach to maintaining user-defined integrity using error detection and correction algorithms similar to those implemented in network protocols, i.e. the Hamming Code Protocol. An implementation of the proposed approach is explained in detail and examines properties of the implementation.

II. PROTECTED DATABASE

There are many ways of securing the database. These ways are based on different aspects of securing the database.

2.1 Authentication of Users

Within this point the author has mentioned about public key encryption (PKI) for the databases that require higher levels of safety one-time passwords X.509 digital certificate smart cards can be used. PKI is very useful when contacting over irrelevant networks like the Internet and both on the internal servers.

2.2 Access control to objects and authentication of authorized applications

In this point means the access control should be defined at the design state. Here main emphasis is given on the roles and based on this access is given to the user.

2.3 Administration policies and procedure

Its mean Security and safety policies with plans are required for varying requirements of data security.

2.4 Secure initial configuration

This point indicates the Policies and procedures also define auditing requirements for securing initial configuration and managing change regulation.

2.5 Auditing

This point refers to auditing the author emphasizes on maintaining logs of the changes to the database management system.

2.6 Backup and recovery strategies

This point refers to the backup and strategies, As the backup and recovery there should be three kinds of backup's cold, hot and logical. All these aspects are traditional and there are vulnerabilities in these security methods which may cause threats to the database system. Henceforth this paper gives detailed information about the vulnerabilities, threats and different security methods to avoid them.

III. PROPOSED SYSTEM

VULNERABILITIES IN DATABASE MANAGEMENT SYSTEM (VDBMS)

Based on our survey conducted the vulnerabilities in the database are defined as: poor architecture, misconfigurations, and vendor bugs incorrect usage.

3.1 Vendor

Bugs refer to buffer overflows and other programming errors that result in users executing the commands they are allowed to be executed. Furthermore Downloading and applying patches usually fix vendor bugs and viruses[6-7].

3.2 Poor architecture

Refers the result of inadequate factoring security into the design of how an application works there. These vulnerabilities are typically the hardest to fix because they require a major rework by the vendor. We can give an example of poor architecture; it would be when a vendor utilizes a weak form of inscription.

3.3 Misconfigurations

Are caused by not accurately locking down databases. Mostly the configuration options of databases can be set in a way that compromises security and safety for that database. Some of these parameters are concluded insecurely by default. But mostly it is not a problem unless you unsuspectingly change the configuration and setting[8-9].

3.4 Incorrect

Usage means to building applications utilizing developer tools in ways that can be used to break into a database. SQL injection is an example of incorrect usage for developer. The authors Marco Vieira and Henrique Madeira have defined that the vulnerabilities in DBMS are an internal factor related to the set of security mechanisms available or not available in the database, the correct configuration of those mechanisms (it is a responsibility of the DBA), and the hidden flaws on the system configuration.

3.5 Irresponsible DBA:

Refers to deactivation of the necessary security mechanisms such as user privileges, authentication, auditing, data encryption which allows intruders to find a way to getaccess to the data into the database.

3.6 Incorrect configuration:

Permits unauthorized users or hackers to access the data in our system.

3.7 Hidden flaws in the database:

May allow hackers to connect to the database server by exploring those faults.

3.8 Unauthorized users:

Means these users "still" the credentials of authorized users in order to access the database server for searching the data.

3.9 Misused Privileges:

Refers to authorized users take advantage of their privileges to maliciously access or destroy our data in a database. Vulnerabilities are also defined by Hassan A. Afyouniin the following manners[10].

IV. METHODOLOGY

Hence as a protection from SQL injection many Intrusion Detection Systems (IDS) have been suggested. A brief description of these IDS is discussed below:

5.1 Misuse Detection System for DBMS

This method has been proposed by Chung *et al.* (1999). It is called a misuse-detectionsystem, created for relational databases. It uses audit data log to retrieve profiles describing typical behaviour of users in Database Management System. The method is present by Lee *et al.* (2000). This method is based on intrusions. Hence this method has used time signatures to discover database intrusions. On the other way similar work was proposed by Low *et al*.(2002).This method is used for Detecting Intrusion in Databases through Fingerprinting Transactions (DIDAFIT).It is a system created using misuse detection approach to show database intrusion detection at the application level in a database.

But another approach towards a database specific intrusion detection mechanism is by Hu and Panda (2003). They proposed and developed a mechanism that is more capable of finding data dependency relationships among transactions and use this information to find hidden anomalies in a database log. Ke Chen *et al.* (2005) developed an intrusion detection model for a database system based on digital amnesty. It gives an additional layer of security against DBMS misuse. On other hand a real-time intrusion detection mechanism based on the profile of user roles has been prescribed by Bertino *et al.* (2005). This total approach is based on mining SQL queries stored in audit log files in a database.

Rietta (2006) described an application layer intrusion detection system, which should take the form of a proxy server and apply an anomaly detection model based on distinct characteristics of SQL and the transaction history of a appropriate user application and user. Aziah Asmawi has proposed SQL Injection and Insider Misuse Detection System (SIIMDS) in 2008 to define both types of intrusions from external and internal threats. Malicious users may access a series of safe information and then apply different techniques to retrieve sensitive data by using that information. To address this inference problems,Yu Chen in has created a semantic inference model (SIM) that symbolize all the possible inference channels from any attribute in the system to the set of elevated sensitive attributes. Hence based on the SIM, the violation detection system keeps track of a user's query history in a database. When a new query is stiffed, all the channels where sensitive information can be stored will be recognizing. If the probability of inferring sensitive information increased a more specified threshold, then the current query request will be revoked. Using the security methods mentioned in section A and B secure and safe database can be created. It may be accessed from anywhere and the security would be managed.

Even though there is no such thing as a 100 percent guarantee in network security, awful obstacles can be placed in the path of SQL injection attack. Anybody of these defenses extremely reduces the chances of a successful SQL injection attack to prevent our data. Implementing all four is a best practice that will supply high degree of protection and safety. Despite its extensive application, your web site does not have to be SQL injection's next suspect. The next section briefs up all the vulnerabilities, threats and security methods of database management system in tabular format which will be beneficial for the development of secure and safe database. There actually is a lot method that web site owners can do to secure against SQL injection attack.

Table 1. Details of VDBMS, TDBMS and SMDBMS

Vulnerabilities (VDBMS)		**Vulnerabilities (VDBMS)**	**Vulnerabilities (VDBMS)**
Vendor Bug	Buffer Overflow, Programming errors	May damage or violate the database	Unauthorized access control policy
Poor Architecture	Weak form of encryption	May damage database environment components (networks, applications, operating systems, DBMS and data)	1.Sorion Security Model 2.Jajodia-Dogan Security Model
Misconfiguration	Not properly locking database	Loss of integrity of the database	1.Physical database integrity protection 2.Logical data integrity protection 3.Data element integrity protection
usage	SQL injection	Misuse of availability of database	Intrusion Detection System like 1. A Misuse Detection System for Database System (DEMIDS) 2.SQL Injection and Insider Misuse Detection System (SIIMDS)

PROPOSED CHECKSUM TECHNIQUE:

5.2. **Data Integrity in an RDBMS**

Data Integrity levels in an RDBMS can be classified into four main kinds, i.e. entity integrity, domain integrity, referential integrity and user-defined integrity.

Entity Integrity:

Useful at the row level in a table, entity integrity ensures that a relation does not have any duplicate rows and that each have a unique primary key that can be defined by one or more of its attributes.

- **Domain integrity**: Values in a column in a table must be drawn from some well-defined "domain" of values. This is the simplest form of an integrity constraint which is maintained at all times and in all circumstances. In effect, a "domain" corresponds to *type* in a conventional programming language, and values in a column must be drawn from one of the available types.

- **Referential Integrity:** This is applied at the table level such that values available in one relation are available and synchronized with those in other relations. Referential integrity is enforced with a primary key and ***F****oreign* ***K****ey* (FK) combination. A foreign key comprises one or more columns in a "child relation" whose values are synchronized with those in the PK in a "parent relation". The FK accepts only those values that exist in the PK in order to maintain the integrity of the relation. Referential integrity is preserved when applying any ***D****ata* ***M****anipulation* ***L****anguage* (DML) operations, i.e. insert, update and delete operations, via the following constraints on the application of such operations:-

1. Restricted: Disallow data modification.
2. Cascaded: Extend the data modification on parent relation to all child relations.
3. Nullified: Set the values of matching FK's in child relation to the value NULL.

- **User-Defined Integrity:**

In effect, user-defined constraints on the ways in which a database can be manipulated. Such constraints are the responsibility of the system administrator, who will administer access rights and enforce rules and regulations. The technique described in the following section is intended to deal with user-defined integrity, a form of integrity typically enforced via triggers and constraints. Our technique is intended to augment conventional approaches to user-defined integrity by exploiting conventional triggers and constraints to ensure that access to critical data is only possible via checksums.

V.PROPOSED ARCHITECTURE

First database contains conventional data of some kind and the second stores associated checksum values, i.e. a result and a remainder, that are associated with critical data in the first database. Figure (1) shows to example databases named "DB1" and "CS1, "DB1" is a conventional database is containing values of any permissible type, e.g. integer, decimal, date including time, etc. The checksum values for critical data in "DB1" are stored in "CS1".

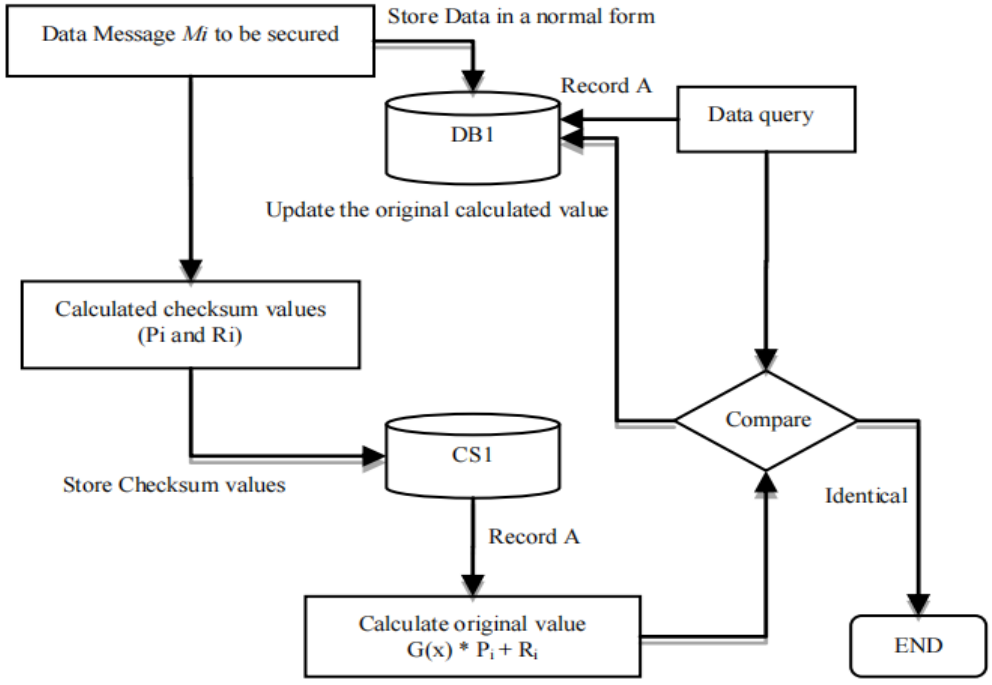

Figure (1)

Checksum values are calculated as the modulus value of a data item (the item being first converted to its ASCII value if necessary) using a divisor chosen by the system administrator. The equation for the recalculation of the original values using the stored checksum related values: Di = G(x) * Pi + Ri ---eq.1

where *Di* is the recalculated value, *G(X)* is a fixed regular divisor chosen, e.g. 1028, *Pi* is the result of the division of *Mi* by *G(x)*, and *Ri* is the modulus of *Mi* and *G(x)* are given by,

Pi = Mi /G(x) ---eq.2

The modulus value is calculated using equation as shown below:

Ri = Mi – (G(X)*(Mi \ G(X))) ---eq.3

Where, *Mi* is assumed the *i th* datum in a database of *N* data sets, \ is the remainder, in this case *Pi* and *Ri* are the checksum values for the *i th* datum in a database of *N* sets of data, were (*N* is *1,2,3,……N*), i.e. By substituting equations (2) and (3) into equation (1) we yield

Di = G(x) * Mi /G(x) + Mi – (G(X)*(Mi \ G(X))) ---eq.4

Equation (4) can be further simplified to give,

Di = 2Mi – (G(X)*(Mi \ G(X))) ---eq.5

V.EXPERIMENTAL RESULTS

For simplicity, it is to be assumed that all access to data in a "DB1" database is protected through the use of an *application view*, and direct access is forbidden by the DBMS's access policy which is set by the database administrator and subject to any other third-party security system, e.g. a Firewall . Application triggers then perform the calculation of checksum values, i.e. result and remainder Pi and Ri respectively and their storage in the "CS1" database.

Consider, next, a customer payment system, composed of several tables, among which are the three tables shown in Figure (3) below.

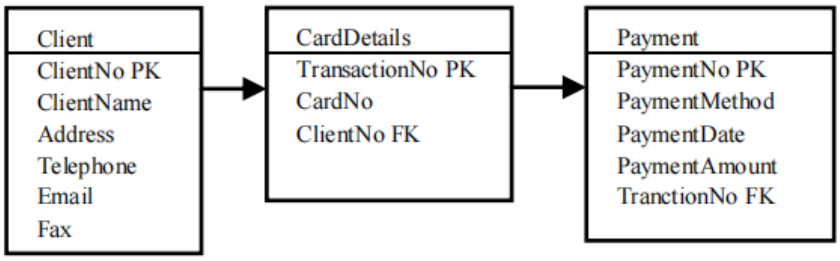

Figure (3) Case Presentation

Our interest here is to preserve the integrity and security of the following fields:

- PaymentAmount Field in the Payment table.
- CardNo Field in the CardDetails table.
- ClientName field in the Client table.

As suggested earlier, direct integrity rules provide basic integrity protection, but are unable to prevent illegal changes to values stored in a database, and similarly unable to prevent the illegal exposure of such values. In such circumstances database transactions, both "legal" and "illegal" will run normally and it will be assumed (even in the event of an illegal transaction) that the integrity of the database is preserved and uncompromised.

6.1 Validation Process

The validation process is concerned with data integrity; this process is carried out by recalculating the data related to the data checksum value using pre-saved indicators related to this checksum, and comparing those values with available data values. The validation algorithm is shown below:

Begin
Identify record in Client table;
Obtain CardNo value for defined record;
Obtain result, remainder values for
defined record;
Compute Checksum
Checksum = result X G(x) + Remainder
Convert checksum to string;
if checksum = 0 and
(CardNo value = 0 or CardNo = null)
{ validate = true;}
else if checksum 0 and CardNo =null
{ validate = false;}
else
{if checksum = CardNo
{validate = true;}
Else
{validate = false;}
}
End

VII. CONCLUSION

An approach that can be used for all types of data; this approach is based on using checksum validation algorithms applied only to critical data such as credit card numbers. This approach protects the data from any illegal changes and guarantees data integrity. In this case the checksum values can be stored in the same database as the regular data or it can be stored in a separate database i.e. "CS1" database, this will increase the level of security. It was also shown that the algorithm provides the necessary data integrity needed for any system regardless of its size and or data type. The proposed data checksum mechanism has an advantage that it can detect whether data has been modified, and in this case the algorithm computes the original data, this will provide a data integrity mechanism to protect the data.

REFERENCES

1) Bertino, E., & Sandhu, R. (2005). Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and secure computing*, *2*(1), 2-19.

2) Ambhore, P. B., Meshram, B. B., & Waghmare, V. B. (2007, August). A implementation of object oriented database security. In *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)* (pp. 359-365). IEEE.

3) Chen, Y., & Chu, W. W. (2008). Protection of database security via collaborative inference detection. In *Intelligence and security informatics* (pp. 275-303). Springer, Berlin, Heidelberg.

4) Silberschatz, A., Korth, H. F., & Sudarshan, S. (1997). *Database system concepts* (Vol. 4). New York: McGraw-Hill.

5) Denning, D. E. (1984, April). Cryptographic checksums for multilevel database security. In *1984 IEEE Symposium on Security and Privacy* (pp. 52-52). IEEE.

6) Andrew, S. (1996). Tanenbaum computer networks. *Computer Networks, Englewood Cliffs*, 141-148.

7) Sivathanu, G., Wright, C. P., & Zadok, E. (2004). *Enhancing file system integrity through checksums*. Technical Report FSL-04-04, Computer Science Department, Stony Brook University.

8) Ajlouni, N., & Ajlouni, F. (2017). Improving DBMS Security through the use of a Checksum Technique. *International Journal of Computer Science and Information Security (IJCSIS)*, *15*(12).

9) Network, A. Oracle® and Open Systems SnapVault®: Backup and Restore for UNIX® P.

10) Sabaratnam, M., Torbjornsen, O., & Hvasshovd, S. O. (1999, December). Cost of ensuring safety in distributed database management systems. In *Proceedings 1999 Pacific Rim International Symposium on Dependable Computing* (pp. 193-200). IEEE.