ORACLE DATABASE SECURITY

Ankur Sharma
Department of Computer Science and Engineering
JIET, Jodhpur
ankur.11jics200@jietjodhpur.ac.in

Sylvester Johnson
Department of Computer Science and Engineering
JIET, Jodhpur
sylvester.10jics120@jietjodhpur.ac.in

ABSTRACT-Database security is a growing concern evidenced by an increase in the number of reported incidents of loss of or unauthorized exposure to sensitive data. As the amount of data collected, retained and shared electronically expands, so does the need to understand database security. Ensuring the security of databases is a complex issue for companies. The more complex the databases are, more complex the security measures that are to be applied. Network and Internet connections to databases may complicate things even further. Also, each and every additional internal user that would be added to user base can create further serious security problems. The purpose of this paper is to highlight and identify the main methods and facets of attack on a database, as well as ways to deflect attacks, through focusing on the delicate issue of data inference. At its core, database security strives to insure that only authenticated users perform authorized activities at authorized times. While database security incorporates a wide array of security topics, not withstanding, physical security, network security, encryption and authentication, this paper focuses on the concepts and mechanisms particular to securing data. Within that context, database security encompasses three constructs: confidentiality or protection of data from unauthorized disclosure, integrity or prevention from unauthorized data access, and availability or the identification of and recovery from hardware and software errors or malicious activity resulting in the denial of data availability.

KEYWORDS: Database security, Attack, Vulnerability, Auditing, Threats, Encryption.

I. INTRODUCTION:-

Databases introduce a number of unique security requirements for their users and administrators. Databases are designed to promote open and flexible access to data but at the same time it is this same open access that makes databases vulnerable to various kinds of malicious activities. One of the main issues faced by database security professionals is avoiding inference capabilities. Basically, inference occurs when users are able to piece together information at one security level to determine a fact that should be protected at a higher security level. These are the some security issues commonly faced by database administrators.

• Unauthorized or unintended activity or misuse by authorized database users, database administrators or systems managers, or by unauthorized users or hackers.

• Malwares' infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the

database, attacks on other systems and the unpredictable failure of database services.

• Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended.

• Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic equipment failures and obsolescence.

• Design flaws and programming bugs in databases and the associated programs and systems, creating various security issues.

• Data corruption or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, criminal damage etc.

II. CLASSICAL ATTACKS:-

The focus of attacks on the company's databases is motivated by the following factors:

• Databases are the mass of information which the company works with;

• Databases can reveal private data by processing public data.

Database security is relative in the next situations:

- Theft and fraud;
- Loss of confidentiality/privacy;
- Loss of privacy;
- Loss of integrity;
- Loss of availability.

The hazards which make these things happen is largely accounted to deliberate human action. Natural type hazards or random events have an impact only on data integrity and availability.

To ensure a minimum security of the databases the following requirements must be satisfied:

- Physical integrity of databases;
- Logical integrity of databases;
- The integrity of each element which composes the database;
- Access control;
- User identification;
- Availability.

The physical and **logical integrity** of databases will require the focus of efforts for protecting the physical integrity of databases, especially the recordings against

destruction. The easiest way to do that is represented by regular backups.

The integrity of each element forming the database will assume that the value of each field may be written or changed only by authorized users and only if there are correct values.

The access control is being done taking into consideration the restrictions of the database administrator. DBMS will apply the security policy of the database administrator (DBA).

This must meet the following requirements:

• **Server security**. Server security involves limiting access to data stored on the server. It is the most important option that has to be taken in consideration and planned carefully.

• **Connections to the database**. Using the ODBC will have to be followed by checking that each connection corresponds to a single user who has access to data.

• **Access control table**. The access control table is the most common form of securing a database. An appropriate use of the table access control involves a close collaboration between the administrator and the base developer.

• **Restriction tables**. Restriction tables will include lists of unsure subjects who could open set-off sessions.

• **Secure IP addresses**. Some servers may be configured to receive only queries from hosts that are in a list. Oracle servers allow blocking queries that are not related to the database.

•**Cancellation of the Server Account**. The ability to suspend an account when "guessing the password" is tried after a predefined number of attempts.

III. DATABASE VULNERABILITY:-

Security breaches are an increasing phenomenon. As more and more databases are made accessible via the Internet and web-based applications, their exposure to security threats will rise. The objective is to reduce susceptibility to these threats. Perhaps the most publicized database application vulnerability has been the SQL injection. SQL injections provide excellent examples for discussing security as they embody one of the most important database security issues, risks inherent to non-validated user input. SQL injections can happen when SQL statements are dynamically created using user input. The threat occurs when users enter malicious code that 'tricks' the database into executing unintended commands. The vulnerability occurs primarily because of the features of the SQL language that allow such things as embedding comments using double hyphens(- -), concatenating SQL statements separated by semicolons, and the ability to query metadata from database data dictionaries. The solution to stopping an SQL injection is input validation. A common example depicts what might occur when a login process is employed on a web page that validates a username and password against data retained in a relational database. The web page provides input forms for user entry of text data. The user-supplied text is used to dynamically create a SQL statement to search the database for matching records. The intention is that valid username and password combinations would be authenticated and the user permitted access to the system. Invalid username and passwords would not be authenticated. However, if a disingenuous user enters malicious text, they could, in essence, gain access to data to which they have no privilege. For instance, the following string, ' OR 1=1 -- entered into the username textbox gains access to the system with knowledge of neither a valid username nor a password. This hack works because the application generates a dynamic query that is formed by concatenating fixed strings with the values entered by the user.

For example, the model SQL code might be:

SELECT Count(*) FROM UsersTable
WHERE UserName = 'contents of username textbox'
AND Password = 'contents of password textbox';

When a user enters a valid username, such as 'Mary' and a password of 'qwerty', the SQL query becomes:

SELECT Count(*) FROM UsersTable
WHERE UserName='Mary'
AND Password='qwerty';

However, if a user enters the following as a username: 'OR 1=1 -- the SQL query becomes:

SELECT Count(*) FROM UsersTable
WHERE UserName='' OR 1=1 - -'
AND Password='';

The expression 1 = 1 is true for every row in the table causing the OR clause to return a value of true. The double hyphens comment out the rest of the SQL query string. This query will return a count greater than zero, assuming there is at least one row in the users table, resulting in what appears to be a successful login. In fact, it is not. Access to the system was successful without a user having to know either a username or password. Another SQL injection is made possible when a database system allows for the processing of stacked queries. Stacked queries are the execution of more than one SQL query in a single function call from an application program. In this case, one string is passed to the database system with multiple queries, each separated by a semicolon. The following example demonstrates a stacked query. The original intent is to allow the user to select attributes of products retained in a Products table. The user injects a stacked query incorporating an additional SQL query that also deletes the Customers table.

*SELECT * FROM PRODUCTS; DROP CUSTOMERS;*

This string when passed as an SQL query will result in the execution of two queries. A listing of all information for all products will be returned. In addition the Customers table will be removed from the database. The table structure will be deleted and all customer data will be lost. In database systems that do not allow stacked queries, or invalidate SQL strings containing a semicolon, this query would not be executed.

IV. AUDITING

Database auditing is used to track database access and user activity. Auditing can be used to identify who accessed database objects, what actions were performed, and what data was changed. Database auditing does not prevent security breaches, but it does provide a way to identify if a breach has occurred. Common categories of database auditing include monitoring database access attempts, Data Control Language (DCL) activities, Data Definition Language (DDL) activities, and Data Manipulation Language (DML) activities [Yang, 2009]. Monitoring access attempts includes retaining information on successful and unsuccessful logon and logoff attempts. DCL audits record changes to user and role privileges, user additions, and user deletions. DDL audits record changes to the database schema such as changes to table structure or attribute datatypes. DML audits record changes to data. In addition, database errors should be monitored [Yang, 2009]. Database auditing is implemented via log files and audit tables. The real challenge of database auditing is deciding what and how much data to retain and how long to keep it. Several options exist. A basic audit trail usually captures user access, system resources used, and changes made to the structure of a database. More complete auditing captures data reads as well as data modifications. An audit trail provides a more complete trace recording of not only user access, but also user actions. This type of facility is included with many database management systems. The most common items that are audited include login attempts, data read and data modifications operations, unsuccessful attempts to access database tables, and attempts to insert data that violates specific constraints.

V. PROTECTING AGAINST DATABASE BYPASS THREATS

Database bypass threats include attacks that target backup media, discarded media, and the operating systems. One of the most widely used technologies used to protect against database bypass threats is encryption. A key milestone in the widespread recognition of encrypt technologies came in 2003 with the passage of California Senate Bill 1386 (SB1386). SB1386 introduced the topic of encryption to a broad audience and since then many other states have passed their own privacy-related laws. Today the need to protect privacy-related information is a global issue as companies expand their operations and businesses. In addition to privacy laws, the Payment Card Industry Data Security Standard (PCI-DSS), first introduced in 2006, has raised awareness across the board for security and the need to render cardholder data unreadable where it is stored and transmitted. While encryption of backup media and proper disposal of media are probably the two most well understood security controls, increasing sophisticated attacks have focused on attacking the servers themselves and gaining access to the raw data files that hold sensitive information.

• Advanced Security TDE

Oracle Advanced Security Transparent Data Encryption (TDE) encrypts and decrypts data through the Oracle database kernel. Data stored on disk is automatically encrypted when loaded into the database and automatically decrypted when users or applications authenticate to the database and pass all database enforced access controls. Plain text data is unavailable when access is attempted at the operating system layer. TDE provides transparent of encryption of data at rest, a requirement today for regulations that range from privacy laws to PCI-DSS as well as protecting against threats directly targeting storage either on production servers or backup media.

TDE enables data owners to maintain control over the data by ensuring that sensitive data does not reside in clear text on the underlying media. Without encryption, operating system attacks leave open the possibility of unimpeded access to sensitive data via direct access at the operating system layer to the files, or disk blocks, that comprise the database, bypassing the authentication and access controls of the database. TDE has distinct advantages over other encryption solutions, in that, a user has to be authenticated to the database and pass authorization checks at both the application and database levels before the data is decrypted. Encryption is especially important when data is moved into cloud environments where the management of storage media and transmission of data is transparent to the data owners themselves. TDE transparently leverages hardware cryptographic acceleration available on Intel® XEON® 5600 and Oracle SPARC® T4 and T5 processors, enabling encryption to be performed with negligible performance overhead in most workload environments. TDE tablespace encryption is integrated with Oracle Compression technologies, enabling storage to be optimized without sacrificing security.

VI. PREVENTING DATABASE BYPASS WITH ENCRYPTION

Data-at-rest encryption is an important control for blocking unauthorized access to sensitive data using methods that circumvent the database. Privileged operating system accounts are just one of the vehicles used by attackers as well as insiders to gain access to sensitive information directly in physical storage.

Oracle Advanced Security Transparent Data Encryption (TDE) stops attackers from bypassing the database and reading sensitive information from storage by encrypting data in the database layer. Applications and users authenticated to the database continue to have access to application data transparently, while unauthenticated users attempting to circumvent the database are denied access to clear text data. To understand this better, consider the fact that privileged operating system users can access database tablespace files and extract sensitive data using simple shell commands. In addition, consider the possibility of attacks that read sensitive data from lost, stolen, or improperly decommissioned disks or backups.

• Oracle Advanced Security Transparent Data Encryption

Transparent Data Encryption resides at an optimal layer within the database to prevent database bypass while still being transparent to applications and easy to deploy. TDE can encrypt individual application table columns or entire application tablespaces. It is transparent to applications because the encryption and decryption processes do not require any application changes, and the application users

do not have to directly deal with encrypted data. Most importantly, TDE's built-in two-tier encryption key management provides full key lifecycle management, tracking the keys across their lifetime with helpful meta-data attributes, and assisted encryption key rotation, switching to a new master key with no downtime. TDE is unique when compared to other alternatives that encrypt entire storage volumes or require new toolkits and programming APIs. Alternative approaches do not protect against many bypass attacks, may require significant application changes, have complex key management, and are not integrated with complementary technologies such as Oracle Advanced Compression, Oracle Recovery Manager, and Oracle Multitenant.

• **Protecting Sensitive Data Using TDE Column Encryption**

Oracle Advanced Security TDE column encryption can be used to encrypt specific data in application tables such as credit card numbers and U.S. Social Security numbers. Customers identify columns within their application schema containing sensitive or regulated data, and then encrypt only those columns. This approach is useful when the database tables are large, only a small number of columns must be encrypted, and the columns are known. TDE column encryption also is useful for warehouse applications where each query is likely to return a very different set of data. Oracle Enterprise Manager Sensitive Data Discovery searches for and identifies sensitive columns quickly. Data encrypted using TDE column encryption remains encrypted on backup media and discarded disk drives, helping prevent unauthorized access and potential data breaches that bypass the database.

• **Protecting Entire Applications Using TDE Tablespace Encryption**

Oracle Advanced Security TDE tablespace encryption protects entire application tables by encrypting the underlying tablespaces. It encrypts application tablespaces regardless of the data's sensitivity and irrespective of its data type. Tablespace encryption simplifies the encryption process because there is no need to identify specific database columns. It is useful when the database contains a large amount of sensitive data to be encrypted and the columns reside in many different locations. TDE tablespace encryption and TDE column encryption can be used independently of one another or together within the same database. As with TDE column encryption, data encrypted with TDE tablespace encryption remains protected on backup media that could pose opportunities for bypass attacks.

CONCLUSION:-

The analysis shows that the Database security presents features that must be taken into account seriously. Transparent Data Encryption encrypts data at rest to stop database bypass attacks from accessing sensitive information in storage. Databases are a favourite target for attackers because of the data these contain and also because of their volume. Data warehousing is the ultimate goal.

Data security and in particular protection of data from unauthorized accesses remain important goals of any data management system. Information security is very important in the present scenario. All well-structured establishments (institutions) entail their own databases having a variety of information pertaining to different contexts. We proposed a simple method to enhance the database security. Even though this is a simple approach it yields strong security preferences when it comes to protecting the data. Database security is becoming an increasingly important topic and students need to develop core understandings in this area. The primary objectives of database security are to prevent unauthorized access to data, prevent unauthorized tampering or modification of data, and to ensure that data remains available when needed. We hope our research work will have a good impact in the database security field.

REFERENCES:-

[1] Defense Information Systems Agency. (2004). *Database security technical implementation guide*, 7(1).
Department of Defense. Retrieved January 31, 2010, from
http://www.databasesecurity.com/dbsec/database-stig-v7r1.pdf

[2] Guimaraes, M. (2006). New challenges in teaching database security. *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development*, Kennesaw, GA, USA, 64-67.

[3] Knox, D. C. (2004). *Effective Oracle database 10g security by design*. New York: McGraw-Hill/Osborne.

[4] Yang, L. 2009. Teaching database security and auditing. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, Chattanooga, TN, USA.

[5] L. Bouganim and Y. Guo, "Database encryption," in Encyclopedia of Cryptography and Security.Springer, 2010, 2nd Edition.

[6] Chen G, Chen K, Dong J (2006) A Database Encryption Scheme for Enhanced Security and Easy Sharing. CSCWD'06, IEEE Proceedings, IEEE Computer Society, Los Alamitos.

[7] Burtescu, E. Database security, in, Knowledge Management. Projects, Systems and Technologies, International Conference, Knowledge Management. Projects, Systems and Technologies, Bucharest, November 9-10, 2006, INFOREC Printing House, Bucharest, 2006

.