**ASSIGNMENT # 2**

Q.1 Define database security and explain database threats?

**DATABASE SECURITY:**

Database security is the use of a wide variety of tools to protect large virtual data storage units. The field is made up of several different components, but is mainly focused on how to best protect user databases from external attacks. Different areas of database security include protecting the data itself (data level security), the applications used to process and store data, the physical servers, and even the network connections that allow users to access databases (system level security).

**DATABASE THREATS:**

**1.Privilege abuse:** When database users are provided with privileges that exceeds their day-to-day job requirement, these privileges may be abused intentionally or unintentionally.

Take, for instance, a database administrator in a financial institution. What will happen if he turns off audit trails or create bogus accounts? He will be able to transfer money from one account to another thereby abusing the excessive privilege intentionally.

Having seen how privilege can be abused intentionally, let us see how privilege can be abused unintentionally. A company is providing a “work from home” option to its employees and the employee takes a backup of sensitive data to work on from his home. This not only violates the security policies of the organization, but also may result in data security breach if the system at home is compromised.

**2. Operating System vulnerabilities:** Vulnerabilities in underlying operating systems like Windows, UNIX, Linux, etc., and the services that are related to the databases could lead to unauthorized access. This may lead to a Denial of Service (DoS) attack. This could be prevented by updating the operating system related security patches as and when they become available.

**3. Database rootkits:** A database rootkit is a program or a procedure that is hidden inside the database and that provides administrator-level privileges to gain access to the data in the database. These rootkits may even turn off alerts triggered by Intrusion Prevention Systems (IPS). It is possible to install a rootkit only after compromising the underlying operating system. This can be avoided by periodical audit trails, else the presence of the database rootkit may go undetected.

**4. Weak authentication**: Weak authentication models allow attackers to employ strategies such as social engineering and brute force to obtain database login credentials and assume the identity of legitimate database users.

**5. Weak audit trails:** A weak audit logging mechanism in a database server represents a critical risk to an organization especially in retail, financial, healthcare, and other industries with stringent regulatory compliance. Regulations such as PCI, SOX, and HIPAA demand extensive logging of actions to reproduce an event at a later point of time in case of an incident. Logging of sensitive or unusual transactions happening in a database must be done in an automated manner for resolving incidents. Audit trails act as the last line of database defense. Audit trails can detect the existence of a violation that could help trace back the violation to a particular point of time and a particular user.

Q.2 what are database control measures?

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical. Database security is a specialist topic within the broader realms of computer security, information security and risk management.

Four main control measures are used to provide security of data in databases:

Access control

Inference control

Flow control

Data encryption

A security problem common to computer systems is that of preventing unauthorized persons from accessing the system itself, either to obtain information or to make malicious changes in a portion of the database. The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function, called access control, is handled by creating user accounts and passwords to control the login process by the DBMS.

Q.3 Explain discretionary Access Control and Mandatory Access Control?

**Mandatory Access Control:**

Mandatory access control (MAC) is a security strategy that restricts the ability individual resource owners have to grant or deny access to resource objects in a file system. MAC criteria are defined by the system administrator, strictly enforced by the operating system (OS) or security kernel, and are unable to be altered by end users.

**Discretionary Access Control:**

In discretionary access control (DAC), the owner of the object specifies which subjects can access the object. This model is called discretionary because the control of access is based on the discretion of the owner.

Most operating systems such as all Windows, Linux, and Macintosh and most flavors of Unix are based on DAC models.

Q.4 Define NoSQL characteristics and core types of NoSQL with Proper Example?

a) Key-Value Store

b) Document-based Store

c) Column-based Store

d) Graph-based database

# NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

# Key-Value Store:

# Key-value stores, or key-value databases, implement a simple data model that pairs a unique key with an associated value. Because this model is simple, it can lead to the development of key-value databases, which are extremely performant and highly scalable for session management and caching in web applications. Implementations differ in the way they are oriented to work with RAM, solid-state drives or disk drives.

# Examples include Aerospike, Berkeley DB, MemchacheDB, Redis and Riak

# Document-based Store:

# Document databases, also called document stores, store semi-structured data and descriptions of that data in document format. They allow developers to create and update programs without needing to reference master schema. Use of document databases has increased along with use of JavaScript and the JavaScript Object Notation (JSON), a data interchange format that has gained wide currency among web application developers, although XML and other data formats can be used as well. Document databases are used for content management and mobile application data handling. Couchbase Server, CouchDB, DocumentDB, MarkLogic and MongoDB are examples of document databases

# Column-based Store:

# In column-oriented NoSQL database, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or the definition of the schema. Read and write is done using columns rather than rows.

# For example: To query the titles from a bunch of a million articles will be a painstaking task while using relational databases as it will go over each location to get item titles. On the other hand, with just one disk access, title of all the items can be obtained.

# Graph-based database:

# In a Graph Base NoSQL Database, you will not find the rigid format of SQL or the tables and columns representation, a flexible graphical representation is instead used which is perfect to address scalability concerns. Graph structures are used with edges, nodes and properties which provides index-free adjacency. Data can be easily transformed from one model to the other using a Graph Base NoSQL database.

# Q.5 Define Lossless and Lossy Decomposition

**Lossless Decomposition:**

Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice. The information will not lose from the relation when decomposed. The join would result in the same original relation.

**Lossy Decomposition:**

As the name suggests, when a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.