chapter 24

Database Security

This chapter discusses techniques for securing databases against a variety of threats. It also presents schemes of providing access privileges to authorized users. Some of the security threats to databases—such as SQL Injection—will be presented. At the end of the chapter we also summarize how a commercial RDBMS—specifically, the Oracle system—provides different types of security. We start in Section 24.1 with an introduction to security issues and the threats to databases, and we give an overview of the control measures that are covered in the rest of this chapter. We also comment on the relationship between data security and privacy as it applies to personal information. Section 24.2 discusses the mechanisms used to grant and revoke privileges in relational database systems and in SQL, mechanisms that are often referred to as **discretionary access control**. In Section 24.3, we present an overview of the mechanisms for enforcing multiple levels of security—a particular concern in database system security that is known as **mandatory access control**. Section 24.3 also introduces the more recently developed strategies of **role-based access control**, and label-based and row-based security. Section 24.3 also provides a brief discussion of XML access control. Section 24.4 discusses a major threat to databases called SQL Injection, and discusses some of the proposed preventive measures against it. Section 24.5 briefly discusses the security problem in statistical databases. Section 24.6 introduces the topic of flow control and mentions problems associated with covert channels. Section 24.7 provides a brief summary of encryption and symmetric key and asymmetric (public) key infrastructure schemes. It also discusses digital certificates. Section 24.8 introduces privacy-preserving techniques, and Section 24.9 presents the current challenges to database security. In Section 24.10, we discuss Oracle label-based security. Finally, Section 24.11 summarizes the chapter. Readers who are interested only in basic database security mechanisms will find it sufficient to cover the material in Sections 24.1 and 24.2.

24.1 Introduction to Database Security Issues[1]

24.1.1 Types of Security

Database security is a broad area that addresses many issues, including the following:

- Various legal and ethical issues regarding the right to access certain information—for example, some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons. In the United States, there are numerous laws governing privacy of information.
- Policy issues at the governmental, institutional, or corporate level as to what kinds of information should not be made publicly available—for example, credit ratings and personal medical records.
- System-related issues such as the *system levels* at which various security functions should be enforced—for example, whether a security function should be handled at the physical hardware level, the operating system level, or the DBMS level.
- The need in some organizations to identify multiple *security levels* and to categorize the data and users based on these classifications—for example, top secret, secret, confidential, and unclassified. The security policy of the organization with respect to permitting access to various classifications of data must be enforced.

Threats to Databases. Threats to databases can result in the loss or degradation of some or all of the following commonly accepted security goals: integrity, availability, and confidentiality.

- **Loss of integrity.** Database integrity refers to the requirement that information be protected from improper modification. Modification of data includes creation, insertion, updating, changing the status of data, and deletion. Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts. If the loss of system or data integrity is not corrected, continued use of the contaminated system or corrupted data could result in inaccuracy, fraud, or erroneous decisions.
- **Loss of availability.** Database availability refers to making objects available to a human user or a program to which they have a legitimate right.
- **Loss of confidentiality.** Database confidentiality refers to the protection of data from unauthorized disclosure. The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Act to the jeopardization of national security. Unauthorized, unanticipated, or unintentional disclosure could result in loss of public confidence, embarrassment, or legal action against the organization.

[1]The substantial contribution of Fariborz Farahmand and Bharath Rengarajan to this and subsequent sections in this chapter is much appreciated.

To protect databases against these types of threats, it is common to implement *four kinds of control measures*: access control, inference control, flow control, and encryption. We discuss each of these in this chapter.

In a multiuser database system, the DBMS must provide techniques to enable certain users or user groups to access selected portions of a database without gaining access to the rest of the database. This is particularly important when a large integrated database is to be used by many different users within the same organization. For example, sensitive information such as employee salaries or performance reviews should be kept confidential from most of the database system's users. A DBMS typically includes a **database security and authorization subsystem** that is responsible for ensuring the security of portions of a database against unauthorized access. It is now customary to refer to two types of database security mechanisms:

- **Discretionary security mechanisms.** These are used to grant privileges to users, including the capability to access specific data files, records, or fields in a specified mode (such as read, insert, delete, or update).
- **Mandatory security mechanisms.** These are used to enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization. For example, a typical security policy is to permit users at a certain classification (or clearance) level to see only the data items classified at the user's own (or lower) classification level. An extension of this is *role-based security,* which enforces policies and privileges based on the concept of organizational roles.

We discuss discretionary security in Section 24.2 and mandatory and role-based security in Section 24.3.

24.1.2 Control Measures

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. We discuss access control techniques in Section 24.1.3.

S**tatistical databases** are used to provide statistical information or summaries of values based on various criteria. For example, a database for population statistics

may provide statistics based on age groups, income levels, household size, education levels, and other criteria. Statistical database users such as government statisticians or market research firms are allowed to access the database to retrieve statistical information about a population but not to access the detailed confidential information about specific individuals. Security for statistical databases must ensure that information about individuals cannot be accessed. It is sometimes possible to deduce or infer certain facts concerning individuals from queries that involve only summary statistics on groups; consequently, this must not be permitted either. This problem, called **statistical database security**, is discussed briefly in Section 24.4. The corresponding control measures are called **inference control** measures.

Another security issue is that of **flow control**, which prevents information from flowing in such a way that it reaches unauthorized users. It is discussed in Section 24.6. Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called **covert channels**. We briefly discuss some issues related to covert channels in Section 24.6.1.

A final control measure is **data encryption**, which is used to protect sensitive data (such as credit card numbers) that is transmitted via some type of communications network. Encryption can be used to provide additional protection for sensitive portions of a database as well. The data is **encoded** using some coding algorithm. An unauthorized user who accesses encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher the data. Encrypting techniques that are very difficult to decode without a key have been developed for military applications. Section 24.7 briefly discusses encryption techniques, including popular techniques such as public key encryption, which is heavily used to support Web-based transactions against databases, and digital signatures, which are used in personal communications.

A comprehensive discussion of security in computer systems and databases is outside the scope of this textbook. We give only a brief overview of database security techniques here. The interested reader can refer to several of the references discussed in the Selected Bibliography at the end of this chapter for a more comprehensive discussion.

24.1.3 Database Security and the DBA

As we discussed in Chapter 1, the database administrator (DBA) is the central authority for managing a database system. The DBA's responsibilities include granting privileges to users who need to use the system and classifying users and data in accordance with the policy of the organization. The DBA has a **DBA account** in the DBMS, sometimes called a **system** or **superuser account**, which provides powerful capabilities that are not made available to regular database accounts and users.[2] DBA-privileged commands include commands for granting and revoking privileges

[2]This account is similar to the *root* or *superuser* accounts that are given to computer system administrators, which allow access to restricted operating system commands.

to individual accounts, users, or user groups and for performing the following types of actions:

1. **Account creation.** This action creates a new account and password for a user or a group of users to enable access to the DBMS.
2. **Privilege granting.** This action permits the DBA to grant certain privileges to certain accounts.
3. **Privilege revocation.** This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.
4. **Security level assignment.** This action consists of assigning user accounts to the appropriate security clearance level.

The DBA is responsible for the overall security of the database system. Action 1 in the preceding list is used to control access to the DBMS as a whole, whereas actions 2 and 3 are used to control *discretionary* database authorization, and action 4 is used to control *mandatory* authorization.

24.1.4 Access Control, User Accounts, and Database Audits

Whenever a person or a group of persons needs to access a database system, the individual or group must first apply for a user account. The DBA will then create a new **account number** and **password** for the user if there is a legitimate need to access the database. The user must **log in** to the DBMS by entering the account number and password whenever database access is needed. The DBMS checks that the account number and password are valid; if they are, the user is permitted to use the DBMS and to access the database. Application programs can also be considered users and are required to log in to the database (see Chapter 13).

It is straightforward to keep track of database users and their accounts and passwords by creating an encrypted table or file with two fields: AccountNumber and Password. This table can easily be maintained by the DBMS. Whenever a new account is created, a new record is inserted into the table. When an account is canceled, the corresponding record must be deleted from the table.

The database system must also keep track of all operations on the database that are applied by a certain user throughout each **login session**, which consists of the sequence of database interactions that a user performs from the time of logging in to the time of logging off. When a user logs in, the DBMS can record the user's account number and associate it with the computer or device from which the user logged in. All operations applied from that computer or device are attributed to the user's account until the user logs off. It is particularly important to keep track of update operations that are applied to the database so that, if the database is tampered with, the DBA can determine which user did the tampering.

To keep a record of all updates applied to the database and of particular users who applied each update, we can modify the *system log*. Recall from Chapters 21 and 23 that the **system log** includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash. We can

expand the log entries so that they also include the account number of the user and the online computer or device ID that applied each operation recorded in the log. If any tampering with the database is suspected, a **database audit** is performed, which consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period. When an illegal or unauthorized operation is found, the DBA can determine the account number used to perform the operation. Database audits are particularly important for sensitive databases that are updated by many transactions and users, such as a banking database that is updated by many bank tellers. A database log that is used mainly for security purposes is sometimes called an **audit trail**.

24.1.5 Sensitive Data and Types of Disclosures

Sensitivity of data is a measure of the importance assigned to the data by its owner, for the purpose of denoting its need for protection. Some databases contain only sensitive data while other databases may contain no sensitive data at all. Handling databases that fall at these two extremes is relatively easy, because these can be covered by access control, which is explained in the next section. The situation becomes tricky when some of the data is sensitive while other data is not.

Several factors can cause data to be classified as sensitive:

1. **Inherently sensitive.** The value of the data itself may be so revealing or confidential that it becomes sensitive—for example, a person's salary or that a patient has HIV/AIDS.
2. **From a sensitive source.** The source of the data may indicate a need for secrecy—for example, an informer whose identity must be kept secret.
3. **Declared sensitive.** The owner of the data may have explicitly declared it as sensitive.
4. A **sensitive attribute or sensitive record.** The particular attribute or record may have been declared sensitive—for example, the salary attribute of an employee or the salary history record in a personnel database.
5. **Sensitive in relation to previously disclosed data.** Some data may not be sensitive by itself but will become sensitive in the presence of some other data—for example, the exact latitude and longitude information for a location where some previously recorded event happened that was later deemed sensitive.

It is the responsibility of the database administrator and security administrator to collectively enforce the security policies of an organization. This dictates whether access should be permitted to a certain database attribute (also known as a *table column* or a *data element*) or not for individual users or for categories of users. Several factors need to be considered before deciding whether it is safe to reveal the data. The three most important factors are data availability, access acceptability, and authenticity assurance.

1. **Data availability.** If a user is updating a field, then this field becomes inaccessible and other users should not be able to view this data. This blocking is

only temporary and only to ensure that no user sees any inaccurate data. This is typically handled by the concurrency control mechanism (see Chapter 22).

2. **Access acceptability.** Data should only be revealed to authorized users. A database administrator may also deny access to a user request even if the request does not directly access a sensitive data item, on the grounds that the requested data may reveal information about the sensitive data that the user is not authorized to have.
3. **Authenticity assurance.** Before granting access, certain external characteristics about the user may also be considered. For example, a user may only be permitted access during working hours. The system may track previous queries to ensure that a combination of queries does not reveal sensitive data. The latter is particularly relevant to statistical database queries (see Section 24.5).

The term *precision*, when used in the security area, refers to allowing as much as possible of the data to be available, subject to protecting exactly the subset of data that is sensitive. The definitions of *security* versus *precision* are as follows:

- **Security:** Means of ensuring that data is kept safe from corruption and that access to it is suitably controlled. To provide security means to disclose only nonsensitive data, and reject any query that references a sensitive field.
- **Precision**: To protect all sensitive data while disclosing as much nonsensitive data as possible.

The ideal combination is to maintain perfect security with maximum precision. If we want to maintain security, some sacrifice has to be made with precision. Hence there is typically a tradeoff between security and precision.

24.1.6 Relationship between Information Security versus Information Privacy

The rapid advancement of the use of information technology (IT) in industry, government, and academia raises challenging questions and problems regarding the protection and use of personal information. Questions of *who* has *what* rights to information about individuals for *which* purposes become more important as we move toward a world in which it is technically possible to know just about anything about anyone.

Deciding how to design privacy considerations in technology for the future includes philosophical, legal, and practical dimensions. There is a considerable overlap between issues related to access to resources (security) and issues related to appropriate use of information (privacy). We now define the difference between *security* versus *privacy*.

Security in information technology refers to many aspects of protecting a system from unauthorized use, including authentication of users, information encryption, access control, firewall policies, and intrusion detection. For our purposes here, we

will limit our treatment of security to the concepts associated with how well a system can protect access to information it contains. The concept of **privacy** goes beyond security. Privacy examines how well the use of personal information that the system acquires about a user conforms to the explicit or implicit assumptions regarding that use. From an end user perspective, privacy can be considered from two different perspectives: *preventing storage* of personal information versus *ensuring appropriate use* of personal information.

For the purposes of this chapter, a simple but useful definition of **privacy** is *the ability of individuals to control the terms under which their personal information is acquired and used*. In summary, security involves technology to ensure that information is appropriately protected. Security is a required building block for privacy to exist. Privacy involves mechanisms to support compliance with some basic principles and other explicitly stated policies. One basic principle is that people should be informed about information collection, told in advance what will be done with their information, and given a reasonable opportunity to approve of such use of the information. A related concept, **trust**, relates to both security and privacy, and is seen as increasing when it is perceived that both security and privacy are provided for.

24.2 Discretionary Access Control Based on Granting and Revoking Privileges

The typical method of enforcing **discretionary access control** in a database system is based on the granting and revoking of **privileges**. Let us consider privileges in the context of a relational DBMS. In particular, we will discuss a system of privileges somewhat similar to the one originally developed for the SQL language (see Chapters 4 and 5). Many current relational DBMSs use some variation of this technique. The main idea is to include statements in the query language that allow the DBA and selected users to grant and revoke privileges.

24.2.1 Types of Discretionary Privileges

In SQL2 and later versions,[3] the concept of an **authorization identifier** is used to refer, roughly speaking, to a user account (or group of user accounts). For simplicity, we will use the words *user* or *account* interchangeably in place of *authorization identifier*. The DBMS must provide selective access to each relation in the database based on specific accounts. Operations may also be controlled; thus, having an account does not necessarily entitle the account holder to all the functionality provided by the DBMS. Informally, there are two levels for assigning privileges to use the database system:

- **The account level.** At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- **The relation (or table) level.** At this level, the DBA can control the privilege to access each individual relation or view in the database.

[3]Discretionary privileges were incorporated into SQL2 and are applicable to later versions of SQL.

The privileges at the **account level** apply to the capabilities provided to the account itself and can include the CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation; the CREATE VIEW privilege; the ALTER privilege, to apply schema changes such as adding or removing attributes from relations; the DROP privilege, to delete relations or views; the MODIFY privilege, to insert, delete, or update tuples; and the SELECT privilege, to retrieve information from the database by using a SELECT query. Notice that these account privileges apply to the account in general. If a certain account does not have the CREATE TABLE privilege, no relations can be created from that account. Account-level privileges *are not* defined as part of SQL2; they are left to the DBMS implementers to define. In earlier versions of SQL, a CREATETAB privilege existed to give an account the privilege to create tables (relations).

The second level of privileges applies to the **relation level**, whether they are base relations or virtual (view) relations. These privileges *are* defined for SQL2. In the following discussion, the term *relation* may refer either to a base relation or to a view, unless we explicitly specify one or the other. Privileges at the relation level specify for each user the individual relations on which each type of command can be applied. Some privileges also refer to individual columns (attributes) of relations. SQL2 commands provide privileges at the *relation and attribute level only.* Although this is quite general, it makes it difficult to create accounts with limited privileges. The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the **access matrix model**, where the rows of a matrix M represent *subjects* (users, accounts, programs) and the columns represent *objects* (relations, records, columns, views, operations). Each position $M(i, j)$ in the matrix represents the types of privileges (read, write, update) that subject i holds on object j.

To control the granting and revoking of relation privileges, each relation R in a database is assigned an **owner account**, which is typically the account that was used when the relation was created in the first place. The owner of a relation is given *all* privileges on that relation. In SQL2, the DBA can assign an owner to a whole schema by creating the schema and associating the appropriate authorization identifier with that schema, using the CREATE SCHEMA command (see Section 4.1.1). The owner account holder can pass privileges on any of the owned relations to other users by **granting** privileges to their accounts. In SQL the following types of privileges can be granted on each individual relation R:

- **SELECT (retrieval or read) privilege on *R*.** Gives the account retrieval privilege. In SQL this gives the account the privilege to use the SELECT statement to retrieve tuples from R.
- **Modification privileges on *R*.** This gives the account the capability to modify the tuples of R. In SQL this includes three privileges: UPDATE, DELETE, and INSERT. These correspond to the three SQL commands (see Section 4.4) for modifying a table R. Additionally, both the INSERT and UPDATE privileges can specify that only certain attributes of R can be modified by the account.

- **References privilege on *R*.** This gives the account the capability to *reference* (or refer to) a relation *R* when specifying integrity constraints. This privilege can also be restricted to specific attributes of *R*.

Notice that to create a view, the account must have the SELECT privilege on *all relations* involved in the view definition in order to specify the query that corresponds to the view.

24.2.2 Specifying Privileges through the Use of Views

The mechanism of **views** is an important *discretionary authorization mechanism* in its own right. For example, if the owner *A* of a relation *R* wants another account *B* to be able to retrieve only some fields of *R,* then *A* can create a view *V* of *R* that includes only those attributes and then grant SELECT on *V* to *B*. The same applies to limiting *B* to retrieving only certain tuples of *R;* a view *V′* can be created by defining the view by means of a query that selects only those tuples from *R* that *A* wants to allow *B* to access. We will illustrate this discussion with the example given in Section 24.2.5.

24.2.3 Revoking of Privileges

In some cases it is desirable to grant a privilege to a user temporarily. For example, the owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed. Hence, a mechanism for **revoking** privileges is needed. In SQL a REVOKE command is included for the purpose of canceling privileges. We will see how the REVOKE command is used in the example in Section 24.2.5.

24.2.4 Propagation of Privileges Using the GRANT OPTION

Whenever the owner *A* of a relation *R* grants a privilege on *R* to another account *B*, the privilege can be given to *B with* or *without* the **GRANT OPTION**. If the GRANT OPTION is given, this means that *B* can also grant that privilege on *R* to other accounts. Suppose that *B* is given the GRANT OPTION by *A* and that *B* then grants the privilege on *R* to a third account *C*, also with the GRANT OPTION. In this way, privileges on *R* can **propagate** to other accounts without the knowledge of the owner of *R*. If the owner account *A* now revokes the privilege granted to *B*, all the privileges that *B* propagated based on that privilege *should automatically be revoked* by the system.

It is possible for a user to receive a certain privilege from two or more sources. For example, A4 may receive a certain UPDATE *R* privilege from *both* A2 and A3. In such a case, if A2 revokes this privilege from A4, A4 will still continue to have the privilege by virtue of having been granted it from A3. If A3 later revokes the privilege from A4, A4 totally loses the privilege. Hence, a DBMS that allows propagation of privileges must keep track of how all the privileges were granted so that revoking of privileges can be done correctly and completely.

24.2.5 An Example to Illustrate Granting and Revoking of Privileges

Suppose that the DBA creates four accounts—A1, A2, A3, and A4—and wants only A1 to be able to create base relations. To do this, the DBA must issue the following GRANT command in SQL:

```
GRANT CREATETAB TO A1;
```

The CREATETAB (create table) privilege gives account A1 the capability to create new database tables (base relations) and is hence an *account privilege.* This privilege was part of earlier versions of SQL but is now left to each individual system implementation to define.

In SQL2 the same effect can be accomplished by having the DBA issue a CREATE SCHEMA command, as follows:

```
CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
```

User account A1 can now create tables under the schema called EXAMPLE. To continue our example, suppose that A1 creates the two base relations EMPLOYEE and DEPARTMENT shown in Figure 24.1; A1 is then the **owner** of these two relations and hence has *all the relation privileges* on each of them.

Next, suppose that account A1 wants to grant to account A2 the privilege to insert and delete tuples in both of these relations. However, A1 does not want A2 to be able to propagate these privileges to additional accounts. A1 can issue the following command:

```
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
```

Notice that the owner account A1 of a relation automatically has the GRANT OPTION, allowing it to grant privileges on the relation to other accounts. However, account A2 cannot grant INSERT and DELETE privileges on the EMPLOYEE and DEPARTMENT tables because A2 was not given the GRANT OPTION in the preceding command.

Next, suppose that A1 wants to allow account A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts. A1 can issue the following command:

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
```

EMPLOYEE

Name	Ssn	Bdate	Address	Sex	Salary	Dno

DEPARTMENT

Dnumber	Dname	Mgr_ssn

Figure 24.1
Schemas for the two relations EMPLOYEE and DEPARTMENT.

The clause WITH GRANT OPTION means that A3 can now propagate the privilege to other accounts by using GRANT. For example, A3 can grant the SELECT privilege on the EMPLOYEE relation to A4 by issuing the following command:

```
GRANT SELECT ON EMPLOYEE TO A4;
```

Notice that A4 cannot propagate the SELECT privilege to other accounts because the GRANT OPTION was not given to A4.

Now suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 then can issue this command:

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

The DBMS must now revoke the SELECT privilege on EMPLOYEE from A3, and it must also *automatically revoke* the SELECT privilege on EMPLOYEE from A4. This is because A3 granted that privilege to A4, but A3 does not have the privilege any more.

Next, suppose that A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. The limitation is to retrieve only the Name, Bdate, and Address attributes and only for the tuples with Dno = 5. A1 then can create the following view:

```
CREATE VIEW A3EMPLOYEE AS
SELECT Name, Bdate, Address
FROM EMPLOYEE
WHERE Dno = 5;
```

After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A3 as follows:

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;
```

Finally, suppose that A1 wants to allow A4 to update only the Salary attribute of EMPLOYEE; A1 can then issue the following command:

```
GRANT UPDATE ON EMPLOYEE (Salary) TO A4;
```

The UPDATE and INSERT privileges can specify particular attributes that may be updated or inserted in a relation. Other privileges (SELECT, DELETE) are not attribute specific, because this specificity can easily be controlled by creating the appropriate views that include only the desired attributes and granting the corresponding privileges on the views. However, because updating views is not always possible (see Chapter 5), the UPDATE and INSERT privileges are given the option to specify the particular attributes of a base relation that may be updated.

24.2.6 Specifying Limits on Propagation of Privileges

Techniques to limit the propagation of privileges have been developed, although they have not yet been implemented in most DBMSs and *are not a part* of SQL. Limiting **horizontal propagation** to an integer number i means that an account B given the GRANT OPTION can grant the privilege to at most i other accounts.