Database Security

11

OBJECTIVES

- Discuss why database security is a serious concern.
- Describe the main database security violations.
- Explain how a database may be protected from security threats using identification, authentication and authorization.
- Discuss a number of techniques for identification and authentication.
- Describe the discretionary access control for authorization.
- Describe SQL support for discretionary access control authorization.
- Discuss mandatory control for authorization.
- Discuss special security concerns for statistical databases.
- Describe the role of the audit policy.
- Explain the basics of security of Internet applications and encryption.
- Introduce the topic of security of outsourced databases.

KEYWORDS

Security, database security, identification, verification, cards, password, user name, authentication, biometrics, discretionary access control, privileges, GRANT, REVOKE, SQL, subjects, mandatory access control, statistical databases, tracker, inference, Internet security, encryption, decryption, symmetrical encryption, asymmetrical encryption, outsourced databases, audit policy.

My grandfather once told me that there were two kinds of people: those who do the work and those who take the credit. He told me to try to be in the first group; there was much less competition there.

Indira Gandhi

11.1 INTRODUCTION

We have noted earlier that a database is a very valuable resource of an enterprise and it is essential that its consistency be maintained at all times. Also, database systems often include valuable enterprise information along with employees' personal information that must be protected from unauthorized retrieval.

The importance of database and computer security may be highlighted by what has been discovered by a number of research teams in UK, Canada and USA during 2008–10. These researchers found that many computers, perhaps several thousands, were compromised by hackers that were based in China. The researchers in Toronto learned what kinds of material had been stolen, including classified assessments about security in several Indian states, and confidential embassy documents about India's relationships in West Africa, Russia and the Middle East. The intruders breached the systems of independent analysts, taking reports on several Indian missile systems. They also obtained a year's worth of Dalai Lama's personal e-mail messages. The reports describing these research are 'Capability of the People's Republic of China to Conduct Cyber Warfare and Computer Network Exploitation' prepared by the US–China Economic and Security Review Commission dated October 9, 2009 (URL for the full 88-page report is http://www.uscc.gov/index.php) and 'Shadows in the Cloud: Investigating Cyber Espionage 2.0' from the University of Toronto (The full report can be downloaded from http://deibert.citizenlab.org/2010/04/new-iwm-report-shadows-in-the-cloud/. A New York Times report dated 6 April 2010 is available at this site http://www.nytimes.com/2010/04/06/science/06cyber.html).

Maintaining database consistency requires many techniques. Some of these have already been discussed in Chapters 9 and 10. For example, in Chapter 9 on concurrency, we discussed techniques for maintaining consistency when a number of transactions are being carried out concurrently. In Chapter 10 on recovery, we discussed how consistency could be maintained in spite of an accidental failure or damage to the database.

We now discuss two further problem areas and solutions to deal with them. These are database security and integrity. Security is discussed in this chapter and integrity is discussed in Chapter 12. A database may become inconsistent as a result of deliberate sabotage of the database system. In addition, the database system needs to be protected against deliberate unauthorized retrieval, corruption or update of data in the database by unauthorized persons. The protection of a database in this context is called *database security*. Also, a database may become inconsistent as a result of errors and mistakes on the part of authorized users. A database system should protect the database from such errors and mistakes. The protection of a database in this context is called maintaining database *integrity*.

In this chapter we discuss database security measures that need to be taken to protect the confidentiality, integrity and availability of the database against intervention by unauthorized persons. Chapter 12 is devoted to a discussion of techniques for maintaining database integrity.

We first define database security violation and database security.

Definition—Database Security Violation
A database security violation takes place when someone carries out an unauthorized retrieval, modification or destruction of information in a database.

Database security involves allowing or disallowing user actions on the database and the objects within it. It is defined as.

Definition—Database Security
Database security is the system, processes and procedures that protect a database from unintended activity including loss of confidentiality, loss of privacy, loss of database integrity or loss of database availability to the users.

Security violations include a number of activities that are listed in the next section. Security is sometimes confused with privacy, a related concept. Privacy, called *information privacy* in the context of databases, deals with the right of individuals to control the flow of information about them. Privacy protection measures will not be discussed here but we note that privacy protection does require that adequate security measures be in place.

This chapter is organized as follows. Security violations and sources of violations in a database system are discussed next. This follows a discussion of various components of database security. Section 11.3 deals with identification and authentication while Section 11.4 deals with authorization which includes discussion on discretionary access control and nondiscretionary control or mandatory control techniques. Security of statistical databases is discussed in Section 11.5. Database audit policy is discussed briefly in Section 11.6 followed by brief discussions on security of internet applications and use of encryption. Finally, a brief discussion on outsourced databases is presented.

11.2 SECURITY VIOLATIONS

As noted earlier, an enterprise database infrastructure is subject to a range of threats. There are a number of ways for classifying database security threats. One approach classifies security violations in the following classes:

1. Loss of Integrity
2. Loss of Availability
3. Loss of Confidentiality

In our approach, security violations may be classified into the more detailed classes presented in Fig. 11.1. Each of these classes may of course be labeled as one of the three classes above (classes 3, 2, 3, 1, 2, and 3 respectively for the six classes in Fig. 11.1).

- Unauthorized disclosure of data (class 3)
- Destruction of data or database (class 2)
- Trojan horse (class 3)
- Corruption or destruction of data (class 1)
- Deliberate interruption of service (class 2)
- Inference (class 3)

Figure 11.1 Security violations in databases

For each class, we present an example.

1. *Unauthorized disclosure of data*—For example, in a university student database, a student looking at assessment records of other students compromises the database security. A lecturer looking at assessment records of students that he/she is not authorized to access also compromises security. A security violation would occur if in a police database, an officer of the department accesses police information of a person to find the address of the person. Also, it has been found that many large organizations assign lower priority to the protection of customer and employee data and a vast majority of security breaches involved confidential customer and employee information.
2. *Destruction of data or database*—A person breaks into a computer system and destroys database files on the computer system.
3. *Trojan horse*—This refers to a transaction that is hidden in another transaction. The hidden transaction becomes active and may breach security when the main transaction is executed. For example, consider a program called *who* that is commonly used by Unix users. A user could put a modified version of *who* in the home directory of the user so that whenever *who* was executed, it not only did what it was supposed to but also copied some of the files into the other user's directory.
4. *Corruption or modification of data*—In a bank database, an employee updates the information in some accounts with a view to embezzling funds. In an interesting example of unauthorized modification of data, an employee of Qantas airlines many years ago managed to regularly modify passenger lists of some flights after the flight had landed. He added his name to these flights and collected millions of frequent flyer points. He was eventually arrested.
5. *Deliberate interruption of service*—A major computer installation is destroyed by a deliberate fire or a bomb thereby destroying a valuable database. As an example, students at a Canadian university in the 1970s destroyed the central computers, interrupting the computer services of the university.
6. *Inference*—A person is able to derive confidential or sensitive information about an individual by accessing information about other individuals or groups from a census database.

These security violations usually lead to either loss of confidentiality, loss of privacy, loss of database integrity or loss of database availability to the users. They can also lead to an enterprise suffering considerable damage.

The motivations behind a deliberate security violation of a computer system are varied. For example, the violator might hope to benefit financially or to cause damage to the enterprise that owns the database perhaps because of hostility towards the enterprise or just wishing to prove that it is possible to penetrate the system security. Whatever the motivation, books and articles on computer crime provide sufficient evidence to support the view that computer security problems are real and extremely important. Computer crime studies have also exposed the vulnerability of some organizations, for whom the loss of their database may result in the collapse of the organization itself.

Sources of Security Threats

Once again, a number of different studies have investigated the sources of database security threats. These studies have shown that in commercial systems the primary threats to database security, in order of importance, are given in Fig. 11.2.

- Errors and omissions
- Dishonest employees
- Fire
- Disgruntled employees
- Water
- Strangers

Figure 11.2 Threats to database security

Trusted insiders therefore remain a significant risk. A majority of organizations have been found to be ineffective in managing the insider

threat and therefore most database fraud or crime is perpetrated by employees of the organization, employees who use their access to the system to commit a crime. One possible solution to insider misuse is careful auditing of databases that have sensitive information in them. Auditing is discussed in Section 11.6. A few years ago, a police employee during the night time tried accessing information about some person in the sensitive database that had information about drug traffic suspects. This database was audited and the employee was caught when someone looked at the auditing data several days later.

Searching for problems that the FBI has with its own employees, I found the following security incidents (for more details, refer to http://www.copwatch.org/TechTV%20%20Top%2010%20List%20of%20Police%20Database%20Abuses%20html.htm):

- Cop Suspected of Using Database to Plan Murder of Ex-wife
- Australian Rookie Cop Checks on 'Potential Girlfriends': 6,900 Database searches in only two months
- FBI Files Sold to mob and international criminals by Nevada Attorney General's Office employee and former FBI agent
- Indiana Police Department banned From FBI database due to misuse
- Prosecutor's office uses database to smear prosecutor's political opponent
- Police Lieutenant charged with abusing database to influence elections

Therefore every database organization has to be very careful in monitoring the security of the database including monitoring of insiders if the information is sensitive. In addition to database auditing, intrusion detection that uses user profiling and data profiling may be used.

In practice, only a small number of problems deal with the remaining security problems like unauthorized access or willful damage to a database system. In spite of the small numbers, the security of a database must be taken seriously.

Computer security concerns can be divided into two broad classes; *internal* security and *external* security. Internal security deals with the operations of the computer system itself and with, for example, access to the system, access to the files, access to networks and inference control. External security on the other hand deals with operations outside the computer system. These, for example, include physical security of the computer server room, the computers that access the database and the rooms in which the computers are placed, security clearance of the personnel accessing the database, security of the network lines, procedures to protect passwords and audit trails. External security problems are beyond the scope of this book.

It should however be made clear at the outset that absolute security is an unrealistic goal. Just as the most well protected banks are sometimes robbed, an adversary with sufficient motivation, resources and ingenuity can compromise the most sophisticated database security safeguards. Also, a disaster of sufficient severity would result in destruction of the database irrespective of the security precautions. An optimum security policy is one in which the cost of implementing protective mechanisms has been balanced against the reduction in risk achieved. The process to achieve a tolerable level of risk at the lowest possible cost is referred to as *risk management*. Furthermore, a DBMS exists to provide flexible and efficient facilities for retrieval, aggregation and manipulation of stored data. Security and integrity controls should be such that an authorized user does not encounter unnecessary problems in accessing the system. This places demands on the system that are often inconsistent with demands to enforce security measures.

While security measures can be costly, experience shows that adequate security is inexpensive compared with the potential consequences of failure to provide adequate protection. Usually an enterprise needs to identify the possible security threats that it might be subjected to and develop a security plan based on those threats. Rare threats should not be ignored if the likely damage due to the threats is very large.

Security policies for different database systems are likely to be different, for example, a bank database, a library database and a police database are likely to have different security concerns and policies. Each security policy would however have a number of components. Some of the components are listed in Fig. 11.3.

1. Identification and authentication
2. Authorization policy
3. Statistical inference policy for statistical databases
4. Encryption
5. Database audit policy

Figure 11.3 Components of database security

Each of the components are now discussed.

11.3 IDENTIFICATION AND AUTHENTICATION

Every database must provide a mechanism that makes sure that the system only allows access to authorized users and each user is allowed to run only those transactions that he/she is authorized to carry out. Therefore a database system must have a comprehensive identification, authentication and authorization mechanism.

Definition—Identification and Authentication

Identification involves a user indicating to the computer system who he/she is while authentication involves the computer system obtaining further information from the user to verify if the user is the person that he/she claims to be.

User identification and authentication are standard means of establishing identity. They are standard operating system problems and most operating system texts provide a discussion on relevant issues.

In general, methods for establishing the identity of an individual can be divided into four classes:

- Knowledge
- Objects
- Actions
- Physiology

Figure 11.4 Four classes of identity verification

1. *Knowledge*—Identification and authentication based on knowledge is the most common form of authentication for computer systems. Common examples of such identification and authentication are username/password and card/PIN number. In username/password authentication, the username may be public but the user is assumed to be the only person who knows the password. The username/password technique does not work particularly well as has been demonstrated by the comparative ease with which hackers have managed to penetrate even so called well secured systems. In one study it was found that passwords were not that difficult to guess as users tend to use passwords that are easy to remember (e.g., same as the username, the user's

name, name of the street the user lives on, name of a relative). When a list of first names, last names, city names, street names and words from a moderate size dictionary were collected it was found that some 86 percent of all passwords on the systems that were checked were on this list. The password authentication can be improved by some simple precautions. For example, it might be possible to check the password a user is using against a dictionary of names and words and not allow common names or words to be used. Some sites are now insisting on an 8-digit password with some numeric or special characters to improve security of passwords.

Username/password authentication continues to be used widely because it is simple and easy to implement.

2. *Objects*—Identification and authentication can be based on objects in the possession of the individual, for example, a card. Each card has a unique identifier stored on it to establish the user's identity. Unfortunately this approach only confirms the identity of the object, not of the user. Objects can be lost, stolen and may be forged allowing a person with someone else's object to gain access. The scheme is considerably improved if the object is combined with some knowledge (e.g., PIN or password). This is the approach being followed by the banks in using a card and a PIN for allowing customers to access ATMs. Even then there have been problems as some people write down their PIN on the card or on a piece of paper. Most ATM cards use a four-digit PIN but many banks now allow use of longer PINs to improve security.
3. *Actions*—It is possible to base authentication on users' actions. For example, handwritten signatures have been used for hundreds of years and the approach has worked quite well. A user may be required to sign on an electronic tablet and the signature may then be compared with the one (or more) stored in the machine. It is also possible to use patterns of computer use behavior once the user is logged on to authenticate the person's identity by comparing the behaviour with the user's profile. These techniques may involve keystroke dynamics (for example, in a 1990 study reliable results were obtained by looking at keystroke latencies when users typed their username and password) and other human-computer interactions.
4. *Physiology*—The most reliable authentication techniques are based on physiology. These are also called biometric techniques and include fingerprints, retinal pattern, face recognition and voice pattern. After the 9/11 terrorist attacks, use of biometrics is becoming more common. The usual approach in biometric techniques is to obtain from the user a sample of the characteristic that is being used and measure this characteristic when the user presents himself/herself for authentication. If the match is close enough then the user is authenticated. Some of the biometric techniques, for example, face recognition, are not particularly reliable. Also, biometrics often requires expensive equipment and are not always suitable for a user wanting to access a database system using a desktop PC.

Most computer systems restrict identification and authentication to the first two categories because of the ease of implementation of the techniques they are based on. Generally, computer systems require a user to identify himself/herself by either typing a login name or by inserting a machine readable card in the client computer. This is then followed by an authentication phase that usually involves the user providing a password or a PIN which is supposed to be known only to the user.

Other authentication schemes are of course possible. For particularly sensitive information, a computer system may use special terminals that are locked or kept in a physically secure location or require the user to carry a machine readable card like a plastic credit card as well as specify login name and password. More recently, inexpensive devices attached to computers are able to read fingerprints of a person for authentication

purposes. Handwritten signature verification is another area in which considerable research has been going on during the last twenty years or so. One bank in Australia is using another approach. For transfer of large amounts of money or for transferring money to a new person, the bank will seek confirmation by sending an SMS message to the account holder's registered mobile phone requesting him/her to type in the code sent via the SMS into the user's computer bank account.

A number of other precautions may also be considered. For example, if appropriate, the system should lock out a user who has failed login on say three successive attempts in a short period of time. Password may be given a lifetime of say three or six months. When a password is changed it may be appropriate to ensure that one of the old passwords is not used.

Furthermore, it should be noted that most computer systems including database systems come with a number of preset default accounts which can lead to security breaches. These default passwords should be changed to secure passwords immediately after the system has been installed.

To conclude the discussion on identification and authentication, the following simple rules should be noted.

(a) To protect an enterprise database strong identification and authentication must be used. In most systems usernames like guest, client, and visitor are often available. All such usernames should be removed from the system. Authentication must use strong passwords with a minimum of eight characters. In addition, users should be encouraged to use lower and upper case characters, as well as numbers, and punctuation marks in their password. Passwords should be regularly changed and each new password should be checked.
(b) System privileges should be allowed only to employees that need to have such privileges.
(c) An account should be locked out after three failed attempts to login to the system.
(d) Unused accounts should be locked or simply deleted after some reasonable amount of time.

11.4 AUTHORIZATION OR ACCESS CONTROL

In contrast to identification and authentication, authorization deals with controlling the type of access that a user is allowed. It may be that we only wish to allow a user by the name *X* to access some parts of the database or employees at level *Y* to some predefined level of access. In addition to access, authorization may also include specification of what a user may do with the access. In the discussion below the users are referred to as *subjects* and the database contents (e.g., tables, views, indexes) are referred to as *objects*.

Definition—Authorization
Database authorization involves allowing certain users to access, process or alter specified parts of the database subject to certain limitations placed on them regarding resources and objects in the database.

Authorization may be based on either subject profiles or object profiles or both:

1. *Subject profiles*—In this scheme the DBMS maintains information on each user and what objects they may access. The information called the *subject profile* enables the DBMS to grant or refuse authorization to carry out a particular transaction.

2. *Object profiles*—In this scheme the DBMS maintains information about each data object. This information called the *object profile* is then used to decide whether a user is allowed access to the objects that he/she wishes to access. The simplest example of object profiles is the file protection mechanism that is provided by most operating systems. For example, in UNIX file management, a file may be readable, writable or executable either by the person whose directory the file is in, or by a group of users specified to the system by the user or by everyone (public). This is not particularly satisfactory as this type of file protection mechanism is very simplistic. In regard to the database, the protection provides only four options as far as the data is concerned:
    - Cannot read or write data
    - Can read but not write data
    - Can read and write data
    - Cannot read but can write data

File protections apply to a file as a whole and therefore different protections for parts of a file are not possible. In the database environment, one may wish to allow selective access to not only parts of the database but also parts of individual tables to a user or a group of users. A technique other than the file protection mechanism described above is then needed.

A database security problem can be compared with a university campus security problem. The database consists of many tables just like a campus consists of many buildings. A simplistic example of security on campus may be something like:

1. People are allowed to enter the campus or not (similar to being allowed to login to a database system).
2. People who are allowed to enter the campus, are either allowed to enter a building (in which case they can enter all rooms in that building) or not allowed to enter the building at all (similar to being able to access a table).

Such simple mechanisms are clearly inadequate in practice and we like to make finer decisions regarding a person's entry to buildings (probably based on each office). Similarly, we would like to make fine-grained decisions on access to a database among different classes of users. In the campus example, students may be allowed to enter the labs and lecture theatres in a building but can enter staff offices only with the permission of the staff member concerned. Locks on each of the offices as well as on the buildings help implement this regulation. Decisions on access to buildings involve things like:

- What day and time of the day it is
- Whether an office has a staff member in it
- Whether the office is locked, and so on

Similar possibilities exist regarding access to a database. For example, a database may allow access to different classes of users on the following basis:

1. All users that have an account may access statistical information from the database.
2. Only users with rating AAA can access and update the personal files of employees.
3. Users with rating AA can access (but not update) the personal files of employees.
4. Users with rating A can access information about employees in their own department.
5. Users that are supervisors may write an assessment of employees they supervise in their departments.

We now provide two techniques that enable fine-grained authorization. These are called *Discretionary Access Control* and *Nondiscretionary* (*or Mandatory*) *Access Control*. Most commercial products currently only implement discretionary access control since discretionary access control (DAC) provides simple and flexible security. However increasingly, perhaps due to the 9/11 terrorism attack in the USA and requirements of the US Department of Defense, database vendors are exploring the possibility of providing mandatory control facilities as well in their products.

11.4.1 Discretionary Access Control (DAC)—Access Matrix Model

A discretionary policy involves providing access to users based on their need-to-access, where the need is determined by someone authorized to determine each user's need. The need may be determined by the user's supervisor or some higher authority in the enterprise that is charged with the responsibility of determining such need and granting and revoking privileges to access and/or update the database as necessary. The database administrator often implements it. In practice in SQL, a user who owns an object (often the database administrator) has the discretion[1] to give permission to others based on their need. The concept of *authorization or access matrix* in a discretionary access control model was proposed by Conway, Maxwell and Morgan in 1972.

Definition—Discretionary Access Control

A discretionary policy involves providing access to users based on their need-to-access, where the need is determined by someone authorized to determine each user's need.

The authorization essentially is a set of triplets (s, o, a) where s is the user, o is the object and a is the action s is authorized to carry out.

The (s, o, a) triples are usually represented by a two-dimensional matrix presenting sets of users, objects and actions (S, O, A) as shown in Fig. 11.5 with columns corresponding to data objects O (not necessarily disjoint) and the rows corresponding to users S may be used to specify access rights A. Each element in the matrix specifies the access rights a user has to a data item. Access rights represent operations performed by users on data items and may include retrieve, insert, delete and update. The data item may be a table, a row or an attribute. Data items may also be defined using views. The model may of course be extended where the access rights might include predicates describing a condition under which the user may access the data item. Discretionary access control therefore is flexible. An example is presented in Fig. 11.5, and allows fine-grained security controls, although not fine-grained enough for some applications, but such controls generally require a person to manage the controls on a day-to-day basis.

	Player	Match	Batting	Bowling
Suresh	–	r	r	r
Anilkumar	r	–	r	–
Saraswati	rw	rw	rw	rw
Inderjit	rwx	rw	–	rw

Figure 11.5 An example of a discretionary matrix

In Fig. 11.5 the symbols r, w and x correspond to the privileges of reading, writing and executing the database object whose name is given at the top of each column. The names of subjects are given in the first column.

1. According to the Oxford Dictionary, discretion means "*the freedom to decide what should be done in a particular situation*". DAC therefore needs someone in the enterprise deciding what privileges should be granted to whom.

This is similar to the use of symbols *r*, *w* and *x* in the UNIX file protection system. This matrix with a large number of objects and a large number of users is often sparse since every user would normally deal with only a small number of objects. Therefore, storing the matrix as a two-dimensional array is not viable.

The matrix may be stored as a table with only non-empty entries or it may be stored either by columns or by rows. A system that stores by columns is commonly known as an access control list (ACL), for example, the UNIX file system is such a system in which each file is accompanied by a list containing subjects and their rights to that file. An implementation that stores by rows is commonly known as a *capability list* in which each subject maintains a list of their rights to objects. The underlying philosophy in discretionary access control is that subjects owning database objects can determine who has access to their database objects, e.g., tables.

Since the authorization matrix can be large, typically DAC supports user groups and object groups which may be hierarchically organized. Once groups are used, it may be necessary to define exceptions. Unfortunately the discretionary access control method can only deal with a coarse granularity of database objects. Also, DAC only deals with users and not processes. A process essentially is given the privileges that are available to the user who generated the process. The process may then execute some malicious programs violating the database security.

Using the SQL GRANT Mechanism

SQL provides facilities for discretionary access control. The owner of a table is automatically granted all privileges to the table. The owner may then grant privileges to other users who in turn may grant privileges to others. Once a user has been granted some privilege by the database system, normally the system would allow that user to grant the same privilege to some other user and the privileges may then be propagated. Of course, if the first user revokes the privilege to the second user, the system should revoke all the privileges that the second user has propagated. The system therefore must not only maintain a table of who has what privilege but also who has granted what privilege and check these every time some privilege is revoked.

It is worth noting that privileges should be granted only when absolutely necessary and should be revoked as soon as the need for a privilege is over.

The granting of privileges may be represented by a graph like the one in Fig. 11.6 below[2].

To simplify the graph, we will only deal with all the privileges on the four tables *A*, *B*, *C* and *D*. Brahma has all privileges on tables *A* and *B*, Vishnu on tables *B* and *C* while Shiva has all privileges on *C* and *D*.

Saraswati has been granted privileges on *A* by Brahma, on *B* by Vishnu and on *C* by Shiva. Lakshmi has been granted privileges on *C* by Vishnu and Parvati has been granted privileges of tables *B* and *C* by Vishnu. Figure 11.6 also shows that the user Hanuman has been granted privileges on table *D* by user Shiva. Rama has been granted privileges on table *A*, Krishna has been granted privileges by three other users, on tables *B* and *C* by Saraswati, on *C* by Lakshmi and on *B* by Parvati.

Now consider what happens if the privileges to Parvati were revoked. She now has no privileges according to our example and as a result Ganesh also loses privilege on *C* but Krishna is not affected since he has privileges on *B* from Saraswati as well. We have assumed that all privileges are of the same type since using privileges for write, read and execute will further complicate the example.

2. In this example, we are using names of users who have been named after some important Hindu gods.

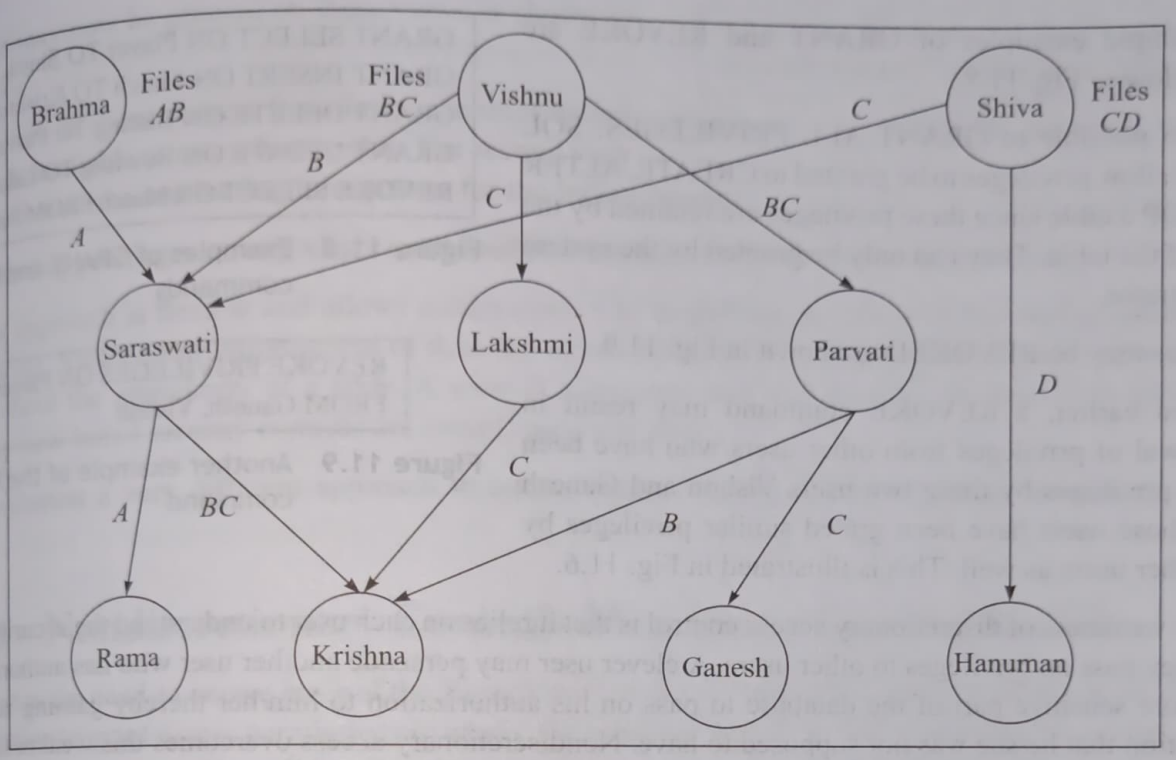

Figure 11.6 Example of an authorization graph

The granularity of many systems is quite coarse as access can be controlled only on the basis of tables. The newer systems are however able to control access as finely as each cell. One may nonetheless wish to have additional security features including value-dependent controls.

A typical SQL GRANT command looks like that shown in Fig. 11.7.

```
GRANT SELECT ON Player, Match
TO Ganesh, Vishnu
```

Figure 11.7 An example of the GRANT command

This grants a privilege to Ganesh and Vishnu to use the SELECT operation on tables named *Player* and *Match*.

A component of the DBMS called the Security Manager deals with authorizations that are issued and also compiles them. It then enforces security based on the rules given by the DBA. It is assumed that the user granting a privilege holds that privilege on the object. SQL allows privileges to be granted for SELECT as well as for other operations, for example:

1. SELECT—This provides rights to read any part of the given table to the user specified.
2. INSERT—This provides rights to the user specified to insert new rows to the given table.
3. DELETE—This provides rights to the user specified to delete rows from the given table.
4. UPDATE—This provides rights to the user specified to update rows in the given table.
5. REFERENCES—This provides the right to the user specified to refer to the table in an integrity constraint.
6. USAGE—This privilege deals with domains of attributes and allows a user to use an element in other users' declarations.
7. TRIGGER—This privilege allows the creation of a trigger on the specified table.
8. EXECUTE—This privilege allows the use of a specified function.

Some simple examples of GRANT and REVOKE are given below in Fig. 11.8.

```
GRANT SELECT ON Player TO Shiva
GRANT INSERT ON Match TO Krishna
GRANT DELETE ON Batting TO Parvati
GRANT UPDATE ON Bowling TO Lakshmi
REVOKE SELECT ON Match FROM Natrajan
```

Figure 11.8 Examples of GRANT and REVOKE commands

It is also possible to GRANT ALL PRIVILEGES. SQL does not allow privileges to be granted to CREATE, ALTER and DROP a table since these privileges are retained by the owner of the table. They can only be granted by the system administrator.

Privileges may be REVOKED as shown in Fig. 11.9.

```
REVOKE PRIVILEGES ON Player
FROM Ganesh, Vishnu
```

Figure 11.9 Another example of the REVOKE command

As noted earlier, a REVOKE command may result in withdrawal of privileges from other users who have been granted privileges by these two users Vishnu and Ganesh unless those users have been grated similar privileges by some other users as well. This is illustrated in Fig. 11.6.

A major weakness of discretionary access control is that it relies on each user to understand the security issues when they pass on privileges to other users. A clever user may persuade another user who has authorizations for a more sensitive part of the database to pass on his authorization to him/her thereby gaining access to information that he/she was not supposed to have. Nondiscretionary access overcomes this weakness by not allowing discretion to any person in the enterprise to grant privileges to others.

11.4.2 Using the SQL View Mechanism

Another useful DAC security mechanism is to provide each user with a tailor-made view (or views) of the database and restrict his/her access to the database through these views. Although the view mechanism allows data to be hidden (including data based on values), we still require control on who is allowed to write into the database through their view. This can be done by using GRANT and REVOKE commands.

Consider the following two views that may be appropriate for some users[3]. Figure 11.10 is a view about Indian Players.

Figure 11.11 presents another view. This is about batting information of Indian players.

```
DEFINE VIEW IPlayers AS
    SELECT PID, FName, LName
    FROM Player
    WHERE Country = 'India'
```

Figure 11.10 A view IPlayers for players from India

```
DEFINE VIEW IBatting AS
    SELECT PID, MID, NRuns
    FROM Batting
    WHERE PID =
        (SELECT PID
        FROM Player
        WHERE Country = 'India')
```

Figure 11.11 A view IBatting for players from India

3. We will continue to use the cricket database although in reality there is no need to control reading access to the tables. We would of course like to ensure that only authorized users can write into them.

Privileges may now be granted on these views as shown in Fig. 11.12.

```
GRANT all privileges on IPlayers to Durga
GRANT SELECT on IBatting to Balaji
```

Figure 11.12 GRANTING privileges on views

The first view is about players from India, the information being made available to users that need it. The second view allows some users access batting information for the Indian players.

A user granting privileges on any views must hold privileges on the tables used in the views.

The view approach is flexible and allows access control to be defined at a level of description close to the requirements. Views allow enforcement of data-dependent policies. It is easy to control read access through views without the granularity of a table. A view is a program and it is possible to control its invocation. However, view based security controls are complicated and slow.

We now discuss a very different approach to authorization, i.e., the nondiscretionary mandatory control model.

11.4.3 Nondiscretionary Control—Mandatory Control Model

In contrast to the need-to-access basis of the discretionary policy, in which the owner of a table or the database has discretion to grant access permissions to others, the nondiscretionary or mandatory access control (MAC) policy involves system-wide policies based on regulations of an enterprise's central authority that do not allow any discretion to any user including table owners and supervisors.

The most commonly used MAC is the multilevel MAC policy, discussed in more detail later. It is based on classifications of both users (called the *subjects*) and objects in the database system. Each subject is then authorized to access all objects that have a classification that is lower or equal to his/her classification.

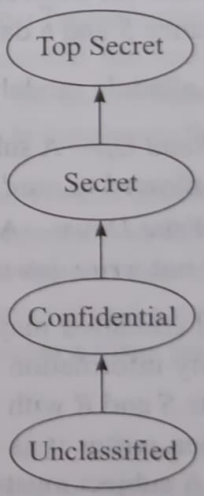

Figure 11.13 A typical classification of database objects

As noted earlier, mandatory control involves assigning each database object to a security *class* and each user or subject a security *clearance*. The security classes may be like those used in the military as shown in Fig. 11.13 and they may range from *unclassified* (anyone can see this) to *confidential* to *secret* and finally *top secret*. Each subject is also assigned a clearance class. Based on these classifications of subjects and objects and a set of rules decided by the enterprise, the DBMS then automatically decides what a subject might or might not be able to do.

Definition—Mandatory Control

Mandatory control involves assigning each database object to a security ***class*** *and each user or subject a security* ***clearance****. Each user is then authorized to access all objects that have a classification that is lower or equal to his/her classification.*

Mandatory control policy is sometimes preferred because it does not provide discretion to anyone to grant privileges to anyone else, in other words a flexibility, which may lead to problems discussed earlier. The mandatory control policy therefore may be considered more reliable. The policy is also called the

Bell–LaPadula model since David Bell and Len LaPadula created the model in 1973 in response to the US Air Force's concerns over the security of time sharing computer systems. It is possible to combine the discretionary access control technique with the mandatory control to enforce additional restrictions, but all users must still follow the basic rules of the mandatory control policies.

Multilevel Bell–LaPadula Model

The *multilevel Bell–LaPadula model* is an example of the mandatory control policy and the model is similar to what is commonly used in security organizations and some industrial organizations. As noted above, basic to such a mandatory security model is a set of *access classes* or *security levels* that represent the classifications associated with the objects in the database as shown in Fig. 11.13 and a set of *clearances* associated with the subjects of the database. Every data object in the database is assigned an access class and each user is assigned a clearance. These clearance classes could be unclassified, confidential, secret and top secret as shown in Fig. 11.13 or they could be denoted as class 1, 2, ...10, where class 1 is the lowest and 10 the highest. For example, class 10 could be top secret in military terms and clearance 10 could be the top secret security clearance.

Bell and LaPadula (BLP) gave a formal model of multilevel security. This model enforces the policy that *information cannot leak to subjects who are not cleared for the information.*

The access classes associated with the data items have a partial ordering associated with them. Let the classes be C_i, $i = 1, 2, \ldots n$ where if for any two classes $C_i > C_j$, then class C_i is said to dominate C_j. To access data in the database, the subject's clearance must dominate the access class of the object. For example, if a subject had a clearance S and a data object has class O then the subject may access the data object as long as $S \geq O$.

The Bell–LaPadula model imposes the following two conditions:

- *No Read Up*—A subject S may read object O only if level(O) ≤ Level(S). As noted above, subjects are not allowed to *read up*.
- *No Write Down*—A subject S may write object O only if level(S) ≤ level(O). In other words, a subject may not *write down*.

The second condition may appear counter-intuitive but the rationale is to ensure that a subject with higher level security information does not write information in a lower level security object. For example, consider two subjects S and R with S having a lower level clearance. If R reads an object O which S is not allowed to read and then writes it as object P at a lower level which S is able to read; sensitive information may leak through P. A subject must never be able to learn information about some highly-labeled object O by reading another low-labeled object P.

A direct implementation of such a system allows the author of a top secret report to retrieve information entered by subjects operating at secret or confidential and merge it with top secret information. The subject with the secret clearance cannot read up to see the top secret result, since the data only flows in one direction between secret and top secret. Some may argue that top secret subjects should be trusted not to do such a thing. Unfortunately, this argument does not take into account the risk of a Trojan horse[4] which we shall now discuss.

4. According to the Oxford Dictionary, a Trojan horse is "*something intended to undermine or secretly overthrow an enemy or opponent*". In the computing context, it is the computer software that appears to perform a desirable function but in fact performs undisclosed malicious functions.

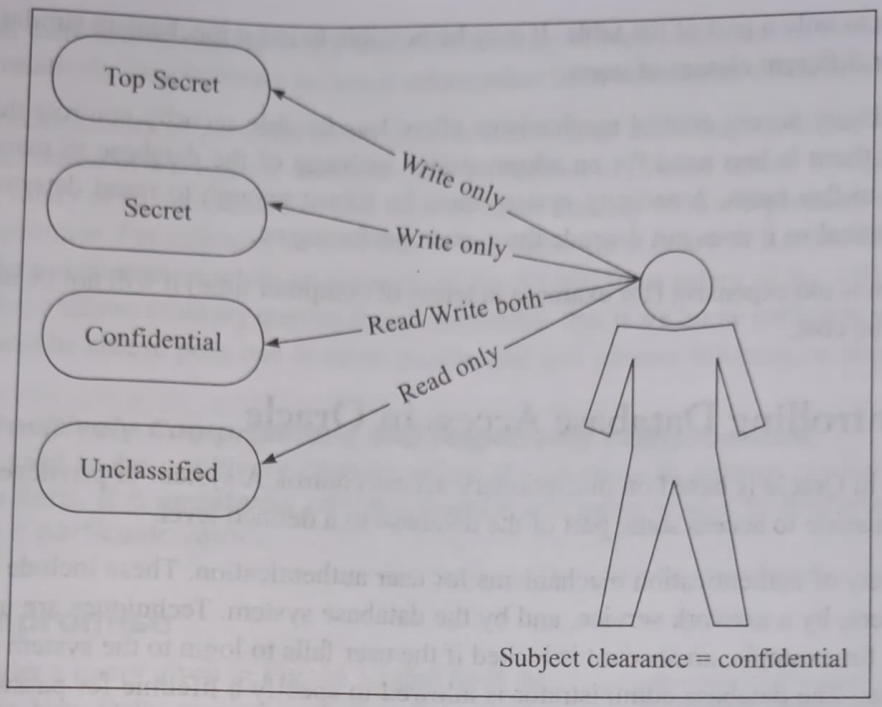

Figure 11.14 Illustrating BLP model rules

For example, consider what would happen if an attacker inserts a macro function with a Trojan horse capability into a word processing file. The attacker has stored the macro function in a confidential file and has told a top secret subject to examine the file. When the subject opens the confidential file, the macro function starts running and copies files from the top secret subject's directory into confidential files belonging to the attacker.

We now discuss weaknesses of the mandatory control model. First, researchers and system developers have found it to be extremely difficult and perhaps impossible to completely prevent information flow between different security levels in the MAC system. Furthermore, it has been reported that the end user community has found a number of cases where the BLP model did not entirely satisfy their operational and security needs. A second serious problem is the virus threat because the MAC system does not prevent a virus introduced at a lower clearance level from propagating into higher clearance levels.

Furthermore, the rules of the Bell–LaPadula model allow 'blind writes'. Subjects can 'write up' and not be able to 'read back' what they have written. Therefore, a subject unsuitable for reading an object is permitted to make changes to that object. One solution to this problem is to insist that writes be restricted to the same security level as the subject allowing a subject to write and read what he/she has written.

A mandatory control mechanism can be implemented in a database in a number of different ways. Perhaps the simplest method would be to treat each table as a data object that is given a security level while each subject is given a clearance level, and to use the rule given above or something similar to control access to the database. Often however this approach is not satisfactory since it does not allow fine-grained security controls. Therefore, it may be necessary to either treat each attribute as a data object with its own security level or treat each row as a data object. The latter would of course result in fairly high security overheads for tables with high cardinality. Nevertheless, a table may contain data of different security classes and it should be possible to specify that a subject who does not have high enough clearance to view the whole table

should have access to only a part of the table. It may be possible to use a mechanism similar to that defining a different view for different classes of users.

In summary, mandatory access control mechanisms allow less flexible security controls than discretionary access control but there is less need for an administrator incharge of the database to manage the security controls on a day-to-day basis. A security system must be robust enough to resist determined attacks but sufficiently economical so it does not degrade the system performance.

If a security system is too expensive (for example in terms of computer time) it will not be used except where the need justifies the cost.

11.4.4 Controlling Database Access in Oracle

Database security in Oracle is based on discretionary access control. A system of privileges is used where a privilege is a permission to access some part of the database to a defined level.

Oracle uses a variety of authentication mechanisms for user authentication. These include authentication by the operating system, by a network service, and by the database system. Techniques are used to control the use of passwords, for example, an account is locked if the user fails to login to the system within a specified number of attempts. The database administrator is allowed to specify a lifetime for passwords, after which passwords expire. Password history is maintained so a user may not reuse a password for a specified amount of time. Passwords are checked for complexity, for example, it is required that a password include an alphabet character, one numeric character, one punctuation mark and differ from the previous password in at least three characters.

In Oracle, a privilege is a right to execute a particular type of SQL statement or to access another user's object. The privileges include the right to retrieve information from another person's table or execute another person's procedure.

A user may grant privileges to other users. Privileges may include system privileges which can only be granted by users that have been granted ADMIN OPTION. There are over 60 system privileges which include privileges such as that to delete a specified row from a specified table in the database.

Oracle also has a concept of role in which a group of privileges may be granted to another user. This is often useful in reducing the security administration work involved in granting similar privileges to many users.

11.5 SECURITY OF STATISTICAL DATABASES

The primary reason for creating statistical databases is to supply s'atistical or aggregate information about groups of individuals to users without revealing confidential information about any individual. For example, a census database maintained by the Bureau of Statistics is designed for use by various types of researchers. Although the census database would have been built from sensitive and confidential information about all citizens, the statistical database normally would have stripped all the personal information from the database and therefore, not be expected to allow any personally identifiable information which can be retrieved. One may think that since a statistical database has no personally identifiable information that may be directly

retrieved by a user, there is little need to worry about the security of such information. We will now show that it is possible and relatively easy to derive personal information from summaries of statistical information.

To preserve the confidentiality of personal information in a statistical database, *statistical inference controls* are required. Compromise of a statistical database must be defined in relative terms. It is generally accepted that an estimate of salary of an individual that is in error by more than say 50% will probably not be considered as a database compromise. For example, an estimate of Rs.15,000 for a salary of Rs. 30,000 could not really be considered to be a compromise while an estimate of Rs. 85,000 for a salary of Rs. 100,000 might be. If the statistical database allows arbitrary queries then it is possible that if we know sufficient information about a person then we will be able to pose one or more queries that will retrieve information about that person.

Definition—Positively Compromised and Negatively Compromised

A database is said to be positively compromised if someone is able to derive a value of a particular data item. It is negativeiy compromised if someone is able to find that a data item does not have a particular value.

Positive Compromise

For example, consider a query given in Fig. 11.15 that finds the number of people in New Delhi earning more than Rupees one crore[5] and the average tax that they pay. We assume the census data is available in a table *Census*.

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Income > 1,00,00,000
AND City = 'New Delhi'
```

Figure 11.15 An example query leading to a positive compromise

Let us assume that the query results in 1 and 750,000, that is, there is only one person in New Delhi with such a high income and the person paid tax of Rs. 750,000. We have therefore derived the tax paid by the only person that qualified. The security of the database and the privacy of that person (whose name is not revealed) therefore have been *compromised*.

Negative Compromise

The query in Fig. 11.16 illustrates that a query may lead to a negative compromise if the result of the query is zero.

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Income > 1,00,00,000
AND City = 'New Delhi'
AND Tax > 10,00,000
```

Figure 11.16 An example query leading to a negative compromise

Let us assume that the query returns a null result. It is now clear that anyone earning more than one crore rupees did not on average pay more than ten lakhs in tax. It does not tell us a lot if the number of people earning that amount is more than, say 10, since we have only managed to find their average tax was above ten lakhs.

Suitable Queries

A commonly suggested solution for this problem of compromising statistical databases is that any query which retrieves less than w rows (or w individuals) should not be allowed. This unfortunately does not work because proceeding as follows would also compromise the database. To illustrate that putting a limit of w

5. This is only a fictitious example. The numbers used have nothing to do with real incomes in India and these numbers may be changed if the reader wishes to do so.

people does not work, let us assume that we believe that only one person in New Delhi has an income of more than one crore. We will now pose two queries that retrieve information about more than w individuals and infer the tax paid by the person earning more than one crore.

We first obtain the number of people in New Delhi and average tax paid by them if they had income of more than ninety lakhs, as shown in Fig. 11.17.

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Income > 90,00,000
AND City = 'New Delhi'
```

Figure 11.17 First query to illustrate that number of rows does not work

Suppose we find that the result is 51 and 500,000. That is, there are 51 people with an income more than 50 lakhs and pay on average a tax of five lakhs. We know that 50 of the 51 individuals are earning below one crore since we suspect only one person has an income of more than one crore.

Suppose we now pose a query to find the number of people and their average tax if their income is more than 90 lakhs and less than one crore. This query is presented in Fig. 11.18.

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Income > 90,000,000
AND Income < 1,00,00,000
AND City = 'New Delhi'
```

Figure 11.18 Second query to illustrate that number of rows does not work

Let us assume that we now find that the result of the query in Fig. 11.18 is 50 and 500,000. That is, there are 50 people with an income greater than 90 lakhs and less than one crore. For convenience we have assumed both average tax figures to be five lakhs, which means the person earning more than one crore also paid tax of only five lakhs.

If the two values of average tax were 5,00,000 and 5,00,100 then we can deduce the tax paid by the one person with income of one crore or more. It works out to be $51 \times 5{,}00{,}000 = 2{,}55{,}00{,}000$ and subtracting from the total tax paid by the 50 people retrieved in query in Fig. 11.18, which is $50 \times 5{,}00{,}100 = 2{,}50{,}05{,}000$), and therefore obtain the figure Rs. 4,95,000.

One may of course suggest other restrictions on the queries, for example, no query should retrieve information about less than w rows or more than $n - w$ rows since it is desirable to exclude queries on a large number of individuals close to the size of the database as well. We assume n is the total number of individuals.

We now define the concept of *suitable queries.*

Definition—Suitable Queries

Queries whose set size falls between (w, n – w) are sometimes called suitable.

Suitable queries also do not prevent compromise as illustrated ahead. We first define what is meant by a *characteristic formula.*

Definition—A characteristic formula

A characteristic formula C is a condition for describing a subgroup of the population (e.g., all people living in New Delhi).

Another term which is used in this context is *query set*. It can be defined as.

Definition—Query Set
The rows that satisfy a characteristic formula are called a query set.

The concept of *tracker* is now defined.

Definition—Tracker
A tracker is a set of auxiliary attributes which are added to the original query.

The auxiliary attributes enable the user to pad the query set so that a query can be formulated to return a result that is larger than that the original query would have returned and the query may be then suitable, and therefore answerable. The effect of the auxiliary characteristics can then be subtracted and the answer to the original query may be obtained. We now define the term *individual tracker*.

Definition—Individual Tracker
A condition like Income > 1,00,00,000 and NOT(City = 'Mumbai') is called an individual tracker.

Generally, if a condition p identifies a single individual in a statistical database and p can be written as $p1$ *and* $p2$ then $p1$ *and not* ($p2$) is called a *tracker* as it lets us track down information on the person we are interested in.

Another Example of Tracking

The basic idea is to find two queries such that one retrieves information about p individuals and another about $p+1$ individuals. Let us assume that an attacker knows that there is a person who has the following attributes:

1. Gender = Male
2. Age < 40
3. Lives in Mumbai
4. Marriage status = Married
5. Income > 1,00,00,000

Let us select the third condition as $p2$ and the remaining four as $p1$. Since we cannot pose a query with the five conditions given above as it results in identifying only one individual, we pose a query ($p1$ and NOT $p2$) as follows.

We should obtain a large number of individuals. Let us assume the result is 1000 and 7,00,000. If we now pose another query which includes all individuals in India, as shown in Fig. 11.20, which does not include the City condition then we should get the above result, plus the person in Mumbai.

Suppose the result now is 1001 and 7,00,015. We can now deduce the tax paid by the one person in Mumbai.

Other approaches have been suggested to overcome such inference problems. One approach suggests adding some random noise to each result but it can be shown that even that approach can lead to a compromise.

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Sex = 'M'
AND Age < 40
AND City <> 'Mumbai'
AND Status = 'Married'
AND Income > 1,00,00,000
```

Figure 11.19 An example of an individual tracker

```
SELECT COUNT(*), AVG(Tax)
FROM Census
WHERE Sex = 'M'
AND Age < 40
AND Status = 'Married'
AND Income > 1,00,00,000
```

Figure 11.20 An example of an individual tracker

We will not discuss compromise in statistical databases any further. A reader interested in further study on this topic may refer to the bibliography at the end of the chapter.

11.6 AUDIT POLICY

An effective tool in database security is the maintenance of an *audit trail*. An audit trail is a record of every transaction that has been executed in a database system. An audit trail usually includes the transaction, the access operations carried out by the transaction and the identification of the user who executed the transaction. Automated recording of at least all sensitive and/or unusual database transactions should be part of any database deployment. Weak database audit policy represents a serious organizational security risk.

In case of suspicion or detection of a database security violation, the audit trail may be analyzed to find the identity of the violator. The knowledge that an audit trail is maintained often deters violators from breaching a system's security. The data generated by an audit trail is very large and effective management of such data is necessary. The large amount of data generated by a busy database system often makes it expensive to analyze the audit trail and this suggests that some method of selective auditing or selective recoding should be considered. We do not discuss these problems any further.

Auditing is generally used for the following:

- Future accountability of current actions on the database.
- Investigations of suspicious activity (for example, some years ago a person working on police computer systems in New Zealand tried to access drug-related records of a friend. He was discovered when a routine audit of the police database activities was carried out).
- Gathering statistics about database activity which can be used for tuning the database.
- Auditing one single user, all the users or a group of users.

Audit records include a variety of information, normally including the following:

(a) *Login, logoff to the database*—The username, date and time of the day, client IP from where the connection is originating. Failed logins also need to be logged and monitored.

(b) *During database usage*—Which application is being used, what information was retrieved, what information was updated? If data was updated, old and new values may need to be saved.

(c) *Data definition usage*—What tables were created and what were dropped, if any? What other DDL commands, if any, were executed?
(d) *Errors*—What errors, if any, occurred during the time the user was logged in?
(e) *Other actions*—Were some integrity constraints changed? Were some security settings changed?
(f) *Highly confidential data*—Was some sensitive data accessed? What database objects were accessed?

Assuming that database auditing is enabled, an audit record including items like those listed above is generated during the execution phase of each statement execution. Often a database audit mechanism on a large and busy database will consume a large amount of CPU and disk resources. The performance decline experienced when an audit is enabled may force an enterprise to scale back or altogether eliminate auditing.

We have only introduced the concepts of database auditing. A reader interested in further information should refer to book by Natan and Afyouni.

11.7 INTERNET APPLICATIONS AND ENCRYPTION

New challenges must be faced when a database is being accessed via the Internet. The Internet is a public network and security risks can expose passwords, accounts, and personal information to unauthorized individuals. In the age of electronic commerce and electronic banking much confidential information, for example, credit card numbers, is communicated via the Internet. Security of such communications is obviously important. Encryption of the data is often used to ensure security. We first define encryption.

Definition—Encryption
Encryption is the process of transforming information using a key and a mathematical algorithm so as the coded information is unintelligible to an unauthorized reader

Encryption is a mathematical algorithm that helps in maintaining secure data in an insecure environment like the Internet. The encryption approach involves scrambling of a message using an encryption algorithm and a key value so that the only person (or device) that can read the message is the device having the corresponding key. Therefore, it cannot be read by an unauthorized person.

The sender transforms given data (called *clear text*) into a new unrecognizable encrypted data set using a secret encryption key which may be created randomly or transformed from a password. The original data cannot be recovered from the encrypted data without the knowledge of a corresponding secret decryption key. Therefore, unless the decryption key is revealed, an adversary who steals the encrypted data cannot succeed in retrieving the unencrypted data. Encryption is most effective in situations like Internet applications where data is communicated online and not physically secured. Indeed, in Internet applications, encryption is the only effective means to prevent data from being stolen as long as the decryption key is not revealed. The decryption key must be conveyed securely to whoever needs to decrypt the data.

Encryption can be symmetric encryption or asymmetric encryption. In symmetric encryption, the data is encrypted and decrypted with the same key which is not revealed to anyone other than the users communicating. In asymmetric encryption, different encryption and decryption keys are used. The encryption key is called

the *public key* and is available to everyone while the decryption key, called the *secret or private key*, is only held by the specific user. The private key is never revealed to anyone and never transmitted over the network.

The process followed in symmetric encryption is illustrated in Fig. 11.21. The sender has some data that is in cleartext form that he/she wishes to send to a receiver. The cleartext is encrypted using a key that is known to both the sender and the receiver. Once the data is encrypted, it is transmitted to the receiver who uses his symmetric key to decrypt the data and obtains the cleartext data that the sender had sent to him/her.

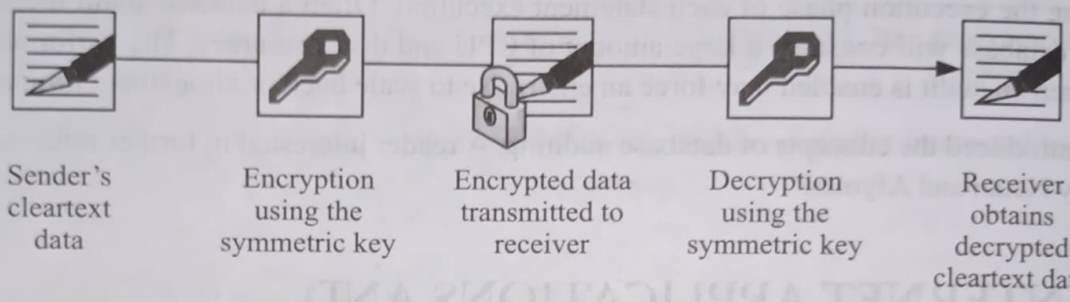

Figure 11.21 An illustration of symmetric encryption

The public/private key algorithms tend to be slow for large amounts of data while the symmetric algorithms are much faster. The public key (often 128-bit long) may be stored on the computer hard disk but the private key needs to be secured carefully, perhaps by storing it after some secure encryption. Communication of keys between the two parties requires careful protocols for both symmetric and asymmetric techniques. Since keys must be communicated via messages, the messages must be communicated in a secure environment, perhaps using encryption. Two issues need to be considered in public/private algorithms. How to communicate the public key and how to communicate the private key? In symmetric algorithms we only need to be concerned about communicating the symmetric key.

We do not discuss the issues of key distribution in this chapter but we illustrate the difficulty of communicating keys by using an example of communicating the public key for a session. Since the key is public, can we communicate it in cleartext? No, since an adversary might be able to intercept such a message and replace the intended key with another public key. The adversary is now able to intercept all messages using the fake public key and decode them!

Implementing a suitable encryption policy should consider the following issues.

(a) *What to encrypt?*—Since encryption involves additional storage costs and performance penalties, it is necessary to identify which part of each table needs to be encrypted. It is best to encrypt selectively.
(b) *Symmetric or Asymmetric?*—Symmetric encryption is faster but asymmetric encryption is more robust and has become the industry standard.
(c) *Key distribution*—A reliable system of key distribution is essential for the success of any encryption system.
(d) *Impact on database performance*—What impact would encryption have on the database performance? Encryption and decryption involve additional processing and therefore there will be some performance degradation.
(e) *Impact on recovery*—How does encryption impact recovery? Could the keys be lost if the system fails? How would key recovery take place?

11.8 SECURITY OF OUTSOURCED DATABASES

One of the major recent developments in the IT industry has been outsourcing. In particular a significant amount of IT work from USA and Europe is now done in countries like India to save cost. Data outsourcing also provides data storage and management. Therefore sensitive data is being managed on remote servers maintained by third party outsourcing vendors in an attempt to save the cost of data management although the service provider may not be fully trustworthy. It has been estimated that the total cost of data management is perhaps five times the cost of initial acquisition of the data.

Database outsourcing clearly poses significant challenges as far as correctness of information returned, confidentiality of information, privacy of personal information stored in the database and transportation via conventional transport or via networks[6] is conserned. In the last few years, this has been demonstrated by cases of data misuse when database management has been outsourced. Clients are therefore reluctant to place sensitive data under the control of a foreign outsourcing company without satisfactory database security assurances. To ensure security of database information, strict security protocols are needed. It may also be desirable to use selective encryption wherever appropriate. If data is encrypted then a user that is allowed to access some parts of the database must be provided with the encryption key. Therefore, different keys must be used for different resources and a user that has access to a number of resources must then manage a number of encryption keys.

We will not discuss these issues any further here. A reader interested in further study on this topic should refer to two chapters given in a recent book edited by Michael Gertz and Sushil Jajodia.

SUMMARY

- There are a number of different ways database security may be violated. Security violation may lead to loss of database integrity, loss of database availability and loss of database confidentiality.
- A list of primary threats to a database system includes dishonest employees, disgruntled employees, fire, etc.
- Some fundamental database security concepts and techniques include identification and authentication.
- The role of authorization or access control is discussed. There are two major access control methods for database systems.
- First discretionary access control method for authorization is discussed. In this approach some senior person in an enterprise has the discretion to give privileges to all employees who themselves may pass to others the privileges they have been given.
- The discretionary access control is possible using the SQL GRANT and REVOKE commands. Some DBMS including Oracle use this approach.
- Views may be used for fine grained database access control.

6. In a 2005 article "Personal Data for 3.9 Million Lost in Transit" the *New York Times* describes one of the largest breaches of data security. A number of other cases of breach of data security have also been reported although most of them appear to involve losing the magnetic media when data was being transported.

- The second access control is called the non-discretionary access control method. In this method, all users and database objects are classified and access is controlled based on this classification. This is similar to military classification of documents as TOP SECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED.
- Non-discretionary control (also called Mandatory Access Control, MAC) is based on the principles of information must not leak to users that are not cleared to have it. It is implemented by imposing the conditions "No read up" and "No write down".
- Security risks also exist in statistical databases and statistical inference controls are required to preserve confidentiality of personal information in statistical data.
- An audit policy that involves keeping records of who is accessing or modifying what data can help discover violation of a database system.
- Given that there is a lot of data that is stored in databases and interchanged via the Internet, it is important to protect security of such databases. Encryption is a widely used technique that scrambles the data when it is being transmitted. Two types of encryption techniques are available, viz., symmetric and asymmetric encryption techniques.
- The growing importance of cloud computing and outsourced databases requires that sensitive databases be protected when databases are stored and managed offsite.

REVIEW QUESTIONS

1. Explain how the security of a database may be compromised. (Section 11.1)
2. What types of security violations may occur in database systems? (Section 11.2)
3. Describe the role of identification and authentication in maintaining database security. (Section 11.3)
4. What is the role of authorization in database security and what techniques of authorization are available? (Section 11.4)
5. Explain authorization table, access control list (ACL) and capability list when used in discretionary access control. (Section 11.4.1)
6. How can the SQL view mechanism and GRANT and REVOKE commands be used in database security? (Sections 11.4 and 11.4.2)
7. What does nondiscretionary or mandatory control technique of authorization try to do? (Section 11.4.3)
8. Explain multilevel Bell–La Padula MAC policy and how it works. (Section 11.4.3)
9. Can statistical databases be compromised? How? Give an example. (Section 11.5)
10. Explain the concept of suitable queries and a tracker in statistical databases. (Section 11.5)
11. Give an example of how an audit policy may help in discovering a security violation in a database? (Section 11.6)
12. Estimate how much audit data will be generated for a bank like ICICI when users access its accounts database.
13. Give an example of a website that requires sensitive information like credit card information and discuss security concerns for it. (Section 11.7)
14. What is symmetric encryption and asymmetric encryption? Which is a better technique? (Section 11.7)
15. Briefly explain security concerns when a company in USA outsources its database applications to India. (Sections 11.8)

SHORT ANSWER QUESTIONS

1. Give an example of a database security violation that you have heard about.
2. List three most common security violations.
3. List three major sources of security threats.
4. What is database user identification?
5. What is database authentication?
6. Give an example illustrating the use of discretionary access control in a database system.
7. Give an example of an access control list?
8. What is an authorization graph?
9. What facilities does SQL provide for database security?
10. Give an example of nondiscretionary control.
11. What are No-Read-Up and No-Write-Down policies?
12. Why is No-Write-Down necessary?
13. Why do we need security controls in statistical databases?
14. What is a negative compromise?
15. Define tracker in statistical databases.
16. Can an audit policy help in protecting a database system against violations?
17. List six items of information that an audit process must save when a user is logged in to a database.
18. When a person buys books from amazon.com, what is the major security concern of the company and the customer?
19. What is meant by encryption and decryption?
20. What is a key in encryption?
21. List two major examples that illustrate security concerns when database applications are outsourced to India.

MULTIPLE CHOICE QUESTIONS

1. Which one of the following is **not** normally a source of database system security threats?
   (a) Dishonest employees
   (b) Dishonest customers
   (c) Disgruntled employees
   (d) Water and fire
2. Which one of the following is correct?
   (a) Identification involves a user indicating to the computer system there identity, or simply who he/she is.
   (b) Authentication and authorization are the same.
   (c) Database authentication involves allowing certain users to access, process or alter specified parts of the database subject to certain limitations.
   (d) None of the above