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 identi verification

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

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 table, a row or an attribute. Data items may also be defined using views. The model may of course be extend where the access rights might include predicates describing a condition under which the user may access t data item. Discretionary access control therefore is flexible. An example is presented in Fig. 11.5, and allo fine-grained security controls, although not fine-grained enough for some applications, but such contr 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 matr

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

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

11.5 SECURITY OF STATISTICAL DATABASES

The primary reason for creating statistical databases is to supply s'atistical or aggregate information ab groups of individuals to users without revealing confidential information about any individual. For examp a census database maintained by the Bureau of Statistics is designed for use by various types of researche Although the census database would have been built from sensitive and confidential information about a citizens, the statistical database normally would have stripped all the personal information from the databas 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 leadin to a negative compromis

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 que which retrieves less than w rows (or w individuals) should not be allowed. This unfortunately does not wo because proceeding as follows would also compromise the database. To illustrate that putting a limit of

5. This is only a fictitious example. The numbers used have nothing to do with real incomes in India and the 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 mo than one crore. We will now pose two queries that retrieve information about more than w individuals an 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 tha number of rows does not wor

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 tha 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).