UNIT 4 SECURITY AND PROTECTION

Structure **Page Nos.**

4.0 INTRODUCTION

Modern organisations depend heavily on their information systems and large investments are made on them annually. These systems are now computerised, and networking has been the common trend during the last decade. The availability of information and computer resources within an organisation as well as between cooperating organisations is often critical for the production of goods and services. In addition, data stored in or derived from the system must be correct, i.e., data integrity must be ensured. In some situations, it is also of great importance to keep information confidential.

Computer security is traditionally defined by the three attributes of confidentiality, integrity, and availability. *Confidentiality* is the prevention of unauthorised disclosure of information. *Integrity* is the prevention of unauthorised modification of information, and *availability* is the prevention of unauthorised withholding of information or resources. Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a subsystem that is malfunctioning. An unprotected resource cannot defend against use (or misuse) by an unauthorised or incompetent user.

Even if a perfectly secure operating system was created, human error (or purposeful human malice) can make it insecure. More importantly, there is no such thing as a completely secure system. No matter how secure the experts might think a particular system is, there exists someone who is clever enough to bypass the security.

It is important to understand that a trade-off always exists between security and the ease of use. It is possible to be *too* secure but security always extracts a price, typically making it more difficult to use your systems for their intended purposes. Users of a secure system need to continually type in, continually change and memorise complex passwords for every computer operation, or use biometric systems with retina and fingerprint and voice scans. All these measures along with confirmations are likely to bring any useful work to a snail’s pace. If the consequences are great enough, then you might have to accept extreme security measures. Security is a relative concept, and gains in security often come only with

penalties in performance. To date, most systems designed to include security in the operating system structure have exhibited either slow response times or awkward user interfaces-or both.

This unit discusses about the various types of security threats and the commonly deployed detective and preventive methods.

4.1 OBJECTIVES

After going through this unit, you should be able to:

- identify the security threats and goals;
- mention the role of security policies and mechanisms;
- discuss the significance of authentication and also describe various types of authentication procedures, and
- describe the various models of protection.

4.2 SECURITY THREATS

System security can mean several things. To have system security we need to protect the system from corruption and we need to protect the data on the system. There are many reasons why these need not be secure.

- Malicious users may try to hack into the system to destroy it.
- Power failure might bring the system down.
- A badly designed system may allow a user to accidentally destroy important data.
- A system may not be able to function any longer because one user fills up the entire disk with garbage.

Although discussions of security usually concentrate on the first of these possibilities, the latter two can be equally damaging the system in practice. One can protect against power failure by using un-interruptible power supplies (UPS). These are units which detect quickly when the power falls below a certain threshold and switch to a battery. Although the battery does not last forever - the UPS gives a system administrator a chance to halt the system by the proper route.

The problem of malicious users has been heightened in recent years by the growth of international networks. Anyone connected to a network can try to log on to almost any machine. If a machine is very insecure, they may succeed. In other words, we are not only looking at our local environment anymore, we must consider potential threats to system security to come from any source. The final point can be controlled by enforcing quotas on how much disk each user is allowed to use.

We can classify the security attacks into two types as mentioned below:

1) **Direct:** This is any direct attack on your specific systems, whether from outside hackers or from disgruntled insiders.

2) **Indirect:** This is general random attack, most commonly computer viruses, computer worms, or computer Trojan horses.

These security attacks make the study of security measures very essential for the following reasons:

- **To prevent loss of data:** You don’t want someone hacking into your system and destroying the work done by you or your team members. Even if you have good back-ups, you still have to identify that the data has been damaged (which can occur at a critical moment when a developer has an immediate need for the damaged data), and then restore the data as best you can from your backup systems.

- **To prevent corruption of data:** Sometimes, the data may not completely be lost, but just be partially corrupted. This can be harder to discover, because unlike complete destruction, there is still data. If the data seems reasonable, you could go a long time before catching the problem, and cascade failure could result in serious problems spreading far and wide through your systems before discovery.
- **To prevent compromise of data:** Sometimes it can be just as bad (or even worse) to have data revealed than to have data destroyed. Imagine the consequences of important trade secrets, corporate plans, financial data, etc. falling in the hands of your competitors.
- **To prevent theft of data:** Some kinds of data are subject to theft. An obvious example is the list of credit card numbers belonging to your customers. Just about anything associated with money can be stolen.
- **To prevent sabotage:** A disgruntled employee, a dishonest competitor, or even a stranger could use any combination of the above activities to maliciously harm your business. Because of the thought and intent, this is the most dangerous kind of attack, the kind that has the potential for the greatest harm to your business.

Let's now discuss some security-related issues that the OS must deal to maintain confidentiality, integrity, and availability of system resources.

4.3 SECURITY POLICIES AND MECHANISMS

Computer systems and especially their protection mechanisms must be penetration resistant. However, most, or perhaps all, current systems have security holes that make them vulnerable. As long as vulnerabilities remain in a system, the security may be circumvented. It is thus not difficult to understand why system owners would like to know how secure their particular system actually is.

Security evaluations have been performed for quite some time by mainly two different methods. The first is to classify systems into a predefined set of categories, each with different security requirements. The other method is referred to as penetration testing, which is a form of stress testing exposing weaknesses in a system. Here, the idea is to let a group of people, normally very skilled in the computer security area, attack a target system.

A **security policy** establishes accountability for information protection by defining a set of rules, conditions, and practices that regulate how an organisation manages, protects, and distributes sensitive information. While substantial effort may be put in by the vendor in implementing the mechanisms to enforce the policy and developing assurance that the mechanisms perform properly, all efforts fail if the policy itself is flawed or poorly understood. For this reason, the standards require that "there must be an explicit and well-defined security policy enforced by the system". A security policy may address confidentiality, integrity, and/or availability.

4.4 AUTHENTICATION

It is the process of verifying a user's identity (who you are) through the use of a shared secret (such as a password), a physical token or an artifact (such as a key or a smart card), or a biometric measure (such as a fingerprint).

These three types of authentication are commonly referred to as something you have (physical token), something you know (shared secret), and something you are (biometric measure).

The types and rigor of authentication methods and technologies vary according to the security requirements or policies associated with specific situations and implementations.

The goal of authentication is to provide "reasonable assurance" that anyone who attempts to access a system or network is a legitimate user. In other words, authentication is designed to limit the possibility that an unauthorised user can gain access by impersonating as an authorised user. Here, again, the organisation's security policy should guide how difficult it is for one user to impersonate another. Highly sensitive or valuable information demands stronger authentication technologies than less sensitive or valuable information.

4.4.1 Passwords

The most common and least stringent form of authentication technology demands that users provide only a valid account name and a password to obtain access to a system or network. The password-based authentication is one-way and normally stores the user-id and password combination in a file that may be stored on the server in an encrypted or plaintext file. Most people using the public e-mail systems use this form of authentication.

Protection of Passwords

Some systems store the passwords of all users in a protected file. However, this scheme is vulnerable to accidental disclosure and to abuse by system administrators. UNIX stores only an encrypted form of every password; a string is the password of user X if its encrypted form matches the encrypted password stored for X.

The encrypted password is accessible to everyone, but one cannot find the clear text password without either guessing it or breaking the encryption algorithm.

Data Encryption Standard (DES)

Encryption is based on one-way functions: functions that are cheap to compute but whose inverse is very expensive to compute. A still widely used, though older encryption algorithm is the Data Encryption Standard (DES), which uses a 56bit key.

UNIX does not encrypt passwords with a secret key, instead, it uses the password as the key to encrypt a standard string. The latter method is not as vulnerable to attacks based on special properties of the "secret" key, which must nevertheless be known to many people.

Limitations of Passwords

Users tend to choose passwords that are easy to guess; persistent guessing attacks can find 80-90% of all passwords. Good guesses include the login name, first names, treat names, words in the on-line dictionary, and any of these reversed or doubled.

The way to defeat these attacks is to choose a password that does not fall into any of those categories. Preferably passwords should contain some uppercase letters, numbers and/or special characters; their presence increases the search space by a large factor.

4.4.2 Alternative Forms of Authentication

Alternative forms of authentication include the following technologies:

- **Biometrics:** These systems read some physical characteristic of the user, such as their fingerprint, facial features, retinal pattern, voiceprints, signature analysis, signature motion analysis, typing rhythm analysis, finger length analysis, DNA analysis. These readings are compared to a database of authorised users to determine identity. The main problems of these schemes are high equipment cost and low acceptance rates.
- **Security Devices or Artifacts:** These systems require use of a special-purpose hardware device that functions like a customised key to gain system access. The device may be inserted into the system like a key or used to generate a code that is then entered into the system. The best example is the use of an ATM card, which is inserted in the machine and also requires

password to be entered simultaneously. It holds the user information on either a magnetic strip or a smart chip capable of processing information.

- **Concentric-ring Authentication:** These systems require users to clear additional authentication hurdles as they access increasingly sensitive information. This approach minimizes the authentication burden as users access less sensitive data while requiring stronger proof of identity for more sensitive resources.

Any authentication method may be broken into, given sufficient time, expense, and knowledge. The goal of authentication technologies is to make entry into system expensive and difficult enough that malicious individuals can be deterred from trying to break the security.

4.5 PROTECTION IN COMPUTER SYSTEMS

The use of computers to store and modify information can simplify the composition, editing, distribution, and reading of messages and documents. These benefits are not free, however, part of the cost is the aggravation of some of the security problems discussed above and the introduction of some new problems as well. Most of the difficulties arise because a computer and its programs are shared amongst several users.

For example, consider a computer program that displays portions of a document on a terminal. The user of such a program is very likely not its developer. It is, in general, possible for the developer to have written the program so that it makes a copy of the displayed information accessible to himself (or a third party) without the permission or knowledge of the user who requested the execution of the program. If the developer is not authorised to view this information, security has been violated.

In compensation for the added complexities automation brings to security, an automated system, if properly constructed, can bestow a number of benefits as well. For example, a computer system can place stricter limits on user discretion. In the paper system, the possessor of a document has complete discretion over its further distribution. An automated system that enforces distribution constraints strictly can prevent the recipient of a message or document from passing it to others. Of course, the recipient can always copy the information by hand or repeat it verbally, but the inability to pass it on directly is a significant barrier.

An automated system can also offer new kinds of access control. Permission to execute certain programs can be granted or denied so that specific operations can be restricted to designated users. Controls can be designed so that some users can execute a program but cannot read or modify it directly. Programs protected in this way might be allowed to access information not directly available to the user, filter it, and pass the results back to the user.

Information contained in an automated system must be protected from three kinds of threats: (1) the *unauthorised disclosure* of information, (2) the *unauthorised modification* of information, and (3) the *unauthorised withholding* of information (usually called *denial of service).*

To protect the computer systems, we often need to apply some security models. Let us see in the next section about the various security models available.

4.6 SECURITY MODELS

Security models are more precise and detailed expressions of security policies discussed as above and are used as guidelines to build and evaluate systems. Models can be discretionary or mandatory. In a *discretionary* model, holders of rights can be

allowed to transfer them at their discretion. In a *mandatory* model only designated roles are allowed to grant rights and users cannot transfer them. These models can be classified into those based on the access matrix, and multilevel models. The first model controls access while the second one attempts to control information flow.

Security Policy vs. Security Model

The Security Policy outlines several high level points: how the data is accessed, the amount of security required, and what the steps are when these requirements are not met. The security model is more in depth and supports the security policy. Security models are an important concept in the design of any security system. They all have different security policies applying to the systems.

4.6.1 Access Matrix Model

The access matrix model for computer protection is based on abstraction of operating system structures. Because of its simplicity and generality, it allows a variety of implementation techniques, as has been widely used.

There are three principal components in the access matrix model:

- A set of passive *objects*,
- A set of active *subjects*, which may manipulate the objects,
- A set of *rules* governing the manipulation of objects by subjects.

Objects are typically files, terminals, devices, and other entities implemented by an operating system. A subject is a process and a *domain* (a set of constraints within which the process may access certain objects). It is important to note that every subject is also an object; thus it may be read or otherwise manipulated by another subject. The ***access matrix*** is a rectangular array with one row per subject and one column per object. The entry for a particular row and column reflects the mode of access between the corresponding subject and object. The mode of access allowed depends on the type of the object and on the functionality of the system; typical modes are read, write, append, and execute.

Objects / Subject	File 1	File 2	File 3
User 1	*r, w*	*R*	*r, w, x*
User 2	*r*	*R*	*r, w, x*
User 3	*r, w, x*	*r, w*	*r, w, x*

Figure 1: An access matrix

All accesses to objects by subjects are subject to some conditions laid down by an enforcement mechanism that refers to the data in the access matrix. This mechanism, called a *reference monitor*, rejects any accesses (including improper attempts to alter the access matrix data) that are not allowed by the current protection state and rules. References to objects of a given type must be validated by the *monitor* for that type. While implementing the access matrix, it has been observed that the access matrix tends to be very sparse if it is implemented as a two-dimensional array. Consequently, implementations that maintain protection of data tend to store them either row wise, keeping with each subject a list of the objects and access modes allowed on it; or column wise, storing with each object a list of those subjects that may access it and the access modes allowed on each. The former approach is called the ***capability list*** approach and the latter is called the ***access control list*** approach.

In general, access control governs each user's ability to read, execute, change, or delete information associated with a particular computer resource. In effect, access control works at two levels: first, to grant or deny the ability to interact with a resource, and second, to control what kinds of operations or activities may be

performed on that resource. Such controls are managed by an access control system. Today, there are numerous methods of access controls implemented or practiced in real-world settings. These include the methods described in the next four sections.

4.6.2 Mandatory Access Control

In a ***Mandatory Access Control (MAC)*** environment, all requests for access to resources are automatically subject to access controls. In such environments, all users and resources are classified and receive one or more security labels (such as "Unclassified," "Secret," and "Top Secret"). When a user requests a resource, the associated security labels are examined and access is permitted only if the user's label is greater than or equal to that of the resource.

4.6.3 Discretionary Access Control

In a ***Discretionary Access Control (DAC)*** environment, resource owners and administrators jointly control access to resources. This model allows for much greater flexibility and drastically reduces the administrative burdens of security implementation.

4.6.4 Rule-Based Access Control

In general, ***rule-based access control*** systems associate explicit access controls with specific system resources, such as files or printers. In such environments, administrators typically establish access rules on a per-resource basis, and the underlying operating system or directory services employ those rules to grant or deny access to users who request access to such resources. Rule-based access controls may use a MAC or DAC scheme, depending on the management role of resource owners.

4.6.5 Role-Based Access Control

Role-based access control (RBAC) enforces access controls depending upon a user's role(s). Roles represent specific organisational duties and are commonly mapped to job titles such as "Administrator," "Developer," or "System Analyst." Obviously, these roles require vastly different network access privileges.

Role definitions and associated access rights must be based upon a thorough understanding of an organisation's ***security policy***. In fact, roles and the access rights that go with them should be directly related to elements of the security policy.

4.6.6 Take-Grant Model

The access matrix model, properly interpreted, corresponds very well to a wide variety of actual computer system implementations. The simplicity of the model, its definition of subjects, objects, and access control mechanisms, is very appealing. Consequently, it has served as the basis for a number of other models and development efforts. We now discuss a model based on access matrix.
Take-grant models use graphs to model access control. They have been well researched. Although explained in the terms of graph theory, these models are fundamentally access matrix models. The graph structure can be represented as an adjacency matrix, and labels on the arcs can be coded as different values in the matrix.

In a take-grant model, the protection state of a system is described by a directed graph that represents the same information found in an access matrix. Nodes in the graph are of two types, one corresponding to subjects and the other to objects. An arc directed from a node A to another node B indicates that the subject (or object) A has some access right(s) to subject (or object) B. The arc is labeled with the set of A's rights to B. The possible access rights are read (r), write (w), take (t), and grant (g). Read and write have the obvious meanings. "*Take*" access implies that node A can take node B's access rights to any other node.

For example, if there is an arc labeled (r, g) from node B to node C, and if the arc from A to B includes a “t” in its label, then an arc from A to C labeled (r, g) could be added to the graph (see *Figure 2*). Conversely, if the arc from A to B is marked with a “g”, B can be granted any access right A possesses. Thus, if A has (w) access to a node D and (g) access to B, an arc from B to D marked (w) can be added to the graph (see *Figure 3*).

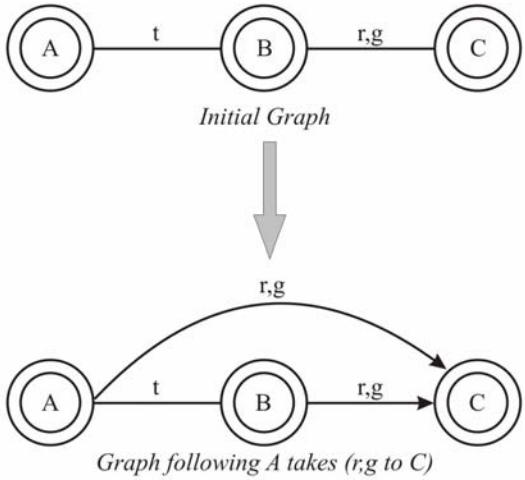

Figure 2: Example of *Take*

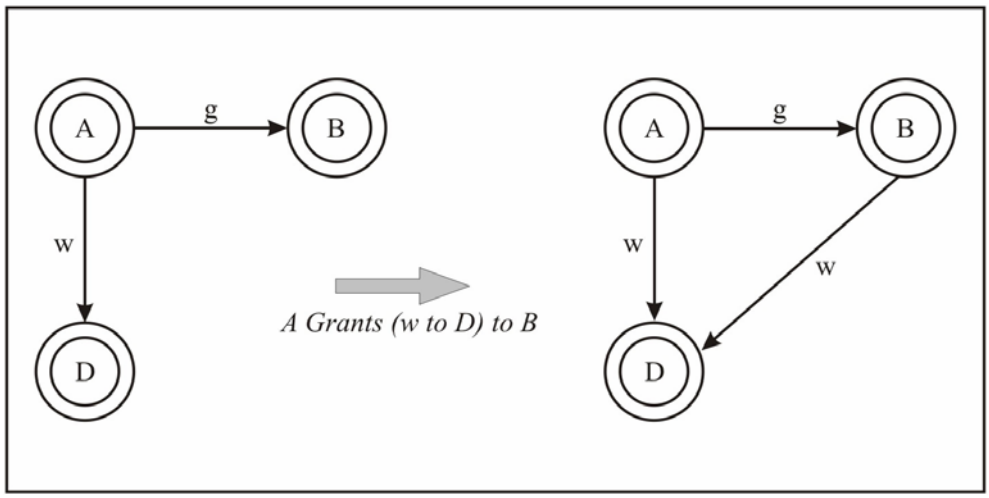

Figure 3: Example of Grant

Because the graph needs only the inclusion of arcs corresponding to non-null entries in the access matrix, it provides a compact way to present the same information given in a relatively sparse access matrix. Capability systems are thus prime candidates for this modeling technique; each arc would then represent a particular capability. Together with the protection graph, the model includes a set of rules for adding and deleting both nodes and arcs to the graph.

Two of these, corresponding to the exercise of “take” and “grant” access rights, have already been described. A “create” rule allows a new node to be added to the graph. If subject A creates a new node Y, both the node Y and an arc AY are added to the graph. The label on AY includes any subset of the possible access rights. A “remove” rule allows an access right to be removed from an arc; if all rights are removed from an arc, the arc is removed as well.

4.6.7 Multilevel Models

This type of models corresponds to the multilevel policies where data is classified into sensitivity levels and users have access according to their clearances. Because of the way they control security they have also been called *data flow* models, they control the allowed flow of data between levels.

- The Bell-La Padula model, intended to control leakage of information between levels.
- The Biba model, which controls the integrity.
- The Lattice model, which generalises the partially ordered levels of the previous models using the concept of mathematical lattices.

Bell and La Padula Model

Bell and La Padula use finite-state machines to formalise their model. They define the various components of the finite state machine, define what it means (formally) for a given system state to be secure, and then consider the transitions that can be allowed so that a secure state can never lead to an insecure state.

In addition to the subjects and objects of the access matrix model, it includes the security levels of the military security system: each subject has a authority and each object has a classification. Each subject also has a *current security level*, which may not exceed the subject's authority. The access matrix is defined as above, and four modes of access are named and specified as follows:

- **Read-only**: subject can read the object but not modify it;
- **Append**: subject can write the object but cannot read it;
- **Execute**: subject can execute the object but cannot read or write it directly; and
- **Read-write**: subject can both read and write the object.

A control attribute, which is like an ownership flag, is also defined. It allows a subject to pass to other subjects some or all of the access modes it possesses for the controlled object. The control attribute itself cannot be passed to other subjects; it is granted to the subject that created the object.

For a system state to be secure, it should hold the following two properties:

(1) The *simple security property:* No subject has read access to any object that has a classification greater than the clearance of the subject; and

(2) The **-property* (pronounced "star-property"): No subject has append-access to an object whose security level is not at least the current security level of the subject; no subject has read-write access to an object whose security level is not equal to the current security level of the subject; and no subject has read access to an object whose security level is not at most the current security level of the subject.

An example restatement of the model is discussed below:

- In this case, we use security levels as (*unclassified* < *confidential* < *secret* < *top-secret*).

The security levels are used to determine appropriate access rights.

The essence of the model can be defined as follows:

- A higher-level subject (e.g., secret level) can always "*read-down"* to the objects with level which is either equal (e.g., secret level) or lower (e.g., confidential / unclassified level). So the system high (top security level in the system) has the read-only access right to all the objects in the entire system.
- A lower-level subject can never "*read-up"* to the higher-level objects, as the model suggests that these objects do not have enough authority to read the high security level information.
- The subject will have the Read-Write access right when the objects are in the same level as the subject.

- A lower-level subject (e.g., confidential level) can always "*write-up*" (**Append** access right) to the objects with level, which is either equal (e.g., confidential level) or higher (e.g., secret / top-secret level). This happens as a result of all the subjects in the higher security levels (e.g., secret / top-secret level) having the authority to read the object from lower levels.
- A higher-level subject can never "*write-down*" to the lower level objects, as the model suggests that these objects are not secure enough to handle the high security level information.

There are also a few problems associated with this model:

- For a subject of a higher-level, all of its information is considered as the same level. So there is no passing of information from this subject to any object in the lower level.
- For an environment where security levels are not hierarchical related, the model does not account for the transfer of information between these domains.
- The model does not allow changes in access permission.
- The model deals only with confidentiality but not on integrity.

Biba Integrity model

This model is a modification of the Bell-La Padula model, with emphasis on the integrity. There are two properties in the Biba model:

- **SI-property** (*simple integrity property*)**:** a subject may have write access to an object only if the security level of the subject is either higher or equals to the level of the object.
- **Integrity property:** a subject has the read-only access to an object *o*, then it can also have the write access to another object *p* only if the security level of *p* is either lower or equals to the level of *o*.

The essence of the model can be defined as follows:

- The read access policy is the same as the Bell-La Padula model.
- No information from a subject can be passed onto an object in a higher security level. This prevents the contamination of the data of higher integrity from the data of lower integrity.

The major problem associated with this model is that there isn't a practical model that can fulfil the confidentiality of the Bell-La Padula model and the integrity of the Biba model.

Information-Flow Models

The significance of the concept of information flow is that it focuses on the actual operations that transfer information between objects. Access control models (such as the original Bell and La Padula model) represent instead the transfer or exercise by subjects of access rights to objects. Information-flow models can be applied to the variables in a program directly, while the access matrix models generally apply to larger objects such as files and processes.

The flow model, compared with the Bell and La Padula model, is relatively uncomplicated. Instead of a series of conditions and properties to be maintained, there is the single requirement that information flow obey the lattice structure as described below. An information-flow model has five components:

- A set of objects, representing information receptacles (e.g., files, program variables, bits),
- A set of processes, representing the active agents responsible for information flow,
- A set of security classes, corresponding to disjoint classes of information,
- An associative and commutative class-combining operator that specifies the class of the information generated by any binary operation on information from two classes, and
- A flow relation that, for any pair of security classes, determines whether information is allowed to flow from one to the other.

Maintaining secure information flow in the modeled system corresponds to ensuring that the actual information that flows between objects do not violate the specified flow relation. This problem is addressed primarily in the context of programming languages. Information flows from an object *x* to an object *y* whenever information stored in *x* is transferred directly to *y* or used to derive information transferred to *y*. Two kinds of information flow, *explicit* and *implicit*, are identified. A flow from *x* to *y* is explicit if the operations causing it are independent of the value of *x* (e.g., in a statement directly assigning the value of *x* to *y*). It is implicit if the statement specifies a flow from some other object *z* to *y*, but execution of the statement depends on the value of *x* (e.g., in the conditional assignment *if x then y: = z;* information about the value of *x* can flow into *y* whether or not the assignment is executed).

The ***lattice model*** for security levels is widely used to describe the structure of military security levels. A lattice is a finite set together with a partial ordering on its elements such that for every pair of elements there is a least upper bound and a greatest lower bound. The simple linear ordering of sensitivity levels has already been defined. Compartment sets can be partially ordered by the subset relation: one compartment set is greater than or equal to another if the latter set is a subset of the former. Classifications, which include a sensitivity level and *a* (perhaps empty) compartment set, can then be partially ordered as follows:

For any sensitivity levels *a*, *b* and any compartment sets *c*, *d*; the relation $(a, c) \geq (b, d)$ exists if and only if $a \geq b$ and $c \supseteq d$. That each pair of classifications has a greatest lower bound and a least upper bound follows from these definitions and the facts that the classification “unclassified, no compartments” is a global lower bound and that we can postulate a classification “top secret, all compartments” as a global upper bound. Because the lattice model matches the military classification structure so closely, it is widely used.

☞ Check Your Progress 1

1) What is computer security and what are the several forms of damages that can be done by the intruders to the computer systems?

   ……

   ……

   ……

2) Explain the role of Access Lists.

   ……

   ……

   ……

4.7 SUMMARY

Security is an important aspect of an OS. The distinction between security and protection is:

Security: Broad sense to refer all concerns about controlled access to facilities.

Protection: Mechanism to support security.

In this unit, we have discussed the concept of security and various threats to it. Also we have discussed various protection and security mechanisms to enforce security to the system.

4.8 SOLUTIONS / ANSWERS

Check Your Progress 1

1) Computer security is required because most organisations can be damaged by hostile software or intruders. There may be several forms of damage which are obviously interrelated. These include:

   - Damage or destruction of computer systems.
   - Damage or destruction of internal data.
   - Loss of sensitive information to hostile parties.
   - Use of sensitive information to steal items of monetary value.
   - Use of sensitive information against the organisation's customers which may result in legal action by customers against the organisation and loss of customers.
   - Damage to the reputation of an organisation.
   - Monetary damage due to loss of sensitive information, destruction of data, hostile use of sensitive data, or damage to the organisation's reputation.

2) The logical place to store authorisation information about a file is with the file itself or with its inode.

   Each element of an Access Control List (ACL) contains a set of user names and a set of operation names. To check whether a given user is allowed to perform a given operation, the kernel searches the list until it finds an entry that applies to the user, and allows the access if the desired operation is named in that entry. The ACL usually ends with a default entry. Systems that use ACLs stop searching when they find an ACL entry that applies to the subject. If this entry denies access to the subject, it does not matter if later entries would also match and grant access; the subject will be denied access. This behaviour is useful for setting up exceptions, such as everyone can access this file except people in this group.

4.9 FURTHER READINGS

1) Abraham Silberschatz and Peter Baer Galvin, *Operating System Concepts*, Wiley and Sons (Asia) Publications, New Delhi.

2) H.M.Deitel, *Operating Systems*, Pearson Education Asia, New Delhi.

3) Andrew S. Tanenbaum, *Operating Systems- Design and implementation*, Prentice Hall of India Pvt. Ltd., New Delhi.

4) Achyut S. Godbole, *Operating Systems*, Second Edition, TMGH, 2005, New Delhi.

5) Milan Milenkovic, *Operating Systems*, TMGH, 2000, New Delhi.

6) D. M. Dhamdhere, *Operating Systems – A Concept Based Approach*, TMGH, 2002, New Delhi.

7) William Stallings, *Operating Systems*, Pearson Education, 2003, New Delhi.