Space Allocation METHODS

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

* Contiguous Allocation
* Linked Allocation
* Indexed Allocation

Contiguous Allocation

* Each file occupies a contiguous address space on disk.
* Assigned disk address is in linear order.
* Easy to implement.
* External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

* Each file carries a list of links to disk blocks.
* Directory contains link / pointer to first block of a file.
* No external fragmentation
* Effectively used in sequential access file.
* Inefficient in case of direct access file.

Indexed Allocation

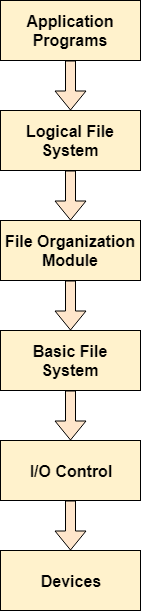
* Provides solutions to problems of contiguous and linked allocation.
* A index block is created having all pointers to files.
* Each file has its own index block which stores the addresses of disk space occupied by the file.
* Directory contains the addresses of index blocks of files.

# File System Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



* When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
* Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.
* Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
* I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.

**FILE SYSTEM IMPLEMENTATION**

Numerous on-disk and in-memory configurations and structures are being used for implementing a file system. These structures differ based on the operating system and the file system but applying some general principles. Here they are portrayed below:

* A boot control block usually contains the information required by the system for booting an operating system from that volume. When the disks do not contain any operating system, this block can be treated as empty. This is typically the first chunk of a volume. In UFS, this is termed as the boot block; in NTFS, it is the partition boot sector.
* A volume control block holds volume or the partition details, such as the number of blocks in the partition, size of the blocks or chunks, free-block count along with free-block pointers. In UFS, it is termed as superblock; in NTFS, it is stored in the master file table.
* A directory structure per file system is required for organizing the files. In UFS, it held the file names and associated 'inode' numbers. In NTFS, it gets stored in the master file table.
* The FCB contains many details regarding any file which includes file permissions, ownership; the size of file and location of data blocks. In UFS, it is called the inode. In NTFS, this information gets stored within the master file table that uses a relational database (RDBM) structure, using a row per file.

# Directory Implementation

There is the number of algorithms by using which, the directories can be implemented. However, the selection of an appropriate directory implementation algorithm may significantly affect the performance of the system.

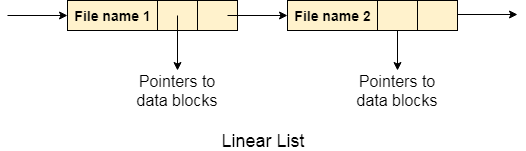
The directory implementation algorithms are classified according to the data structure they are using. There are mainly two algorithms which are used in these days.

### . Linear List

In this algorithm, all the files in a directory are maintained as singly lined list. Each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.

**Characteristics**

1. When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not. In case, it doesn't exist, the file can be created at the beginning or at the end. Therefore, searching for a unique name is a big concern because traversing the whole list takes time.
2. The list needs to be traversed in case of every operation (creation, deletion, updating, etc) on the files therefore the systems become inefficient.

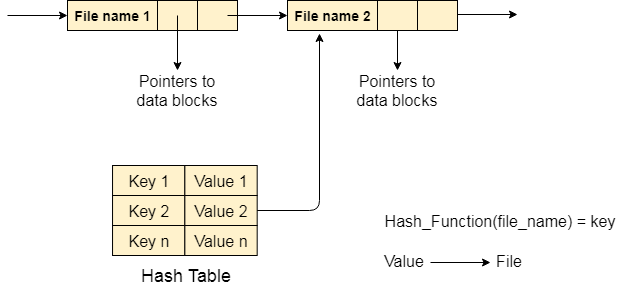


2. Hash Table

To overcome the drawbacks of singly linked list implementation of directories, there is an alternative approach that is hash table. This approach suggests to use hash table along with the linked lists.

A key-value pair for each file in the directory gets generated and stored in the hash table. The key can be determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

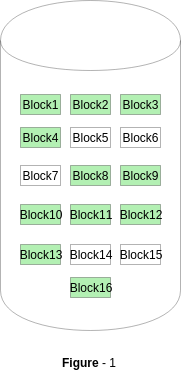
Now, searching becomes efficient due to the fact that now, entire list will not be searched on every operating. Only hash table entries are checked using the key and if an entry found then the corresponding file will be fetched using the value.



# Free space management in Operating System

The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial. The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory. The free space list can be implemented mainly as:

1. **Bitmap or Bit vector –**  
   A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: 0 indicates that the block is allocated and 1 indicates a free block.  
   The given instance of disk blocks on the disk in Figure 1 (where green blocks are allocated) can be represented by a bitmap of 16 bits as:**0000111000000110**.



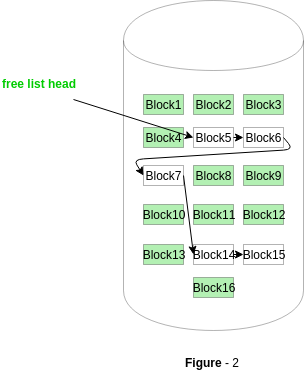
**Advantages –**

* + Simple to understand.
  + Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

The block number can be calculated as:  
*(number of bits per word) \*(number of 0-values words) + offset of bit first bit 1 in the non-zero word*.

For the Figure-1, we scan the bitmap sequentially for the first non-zero word.  
The first group of 8 bits (00001110) constitute a non-zero word since all bits are not 0. After the non-0 word is found, we look for the first 1 bit. This is the 5th bit of the non-zero word. So, offset = 5.  
Therefore, the first free block number = 8\*0+5 = 5.

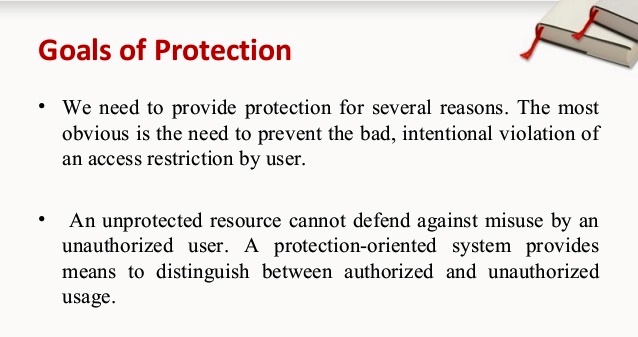
1. **Linked List –**  
   In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

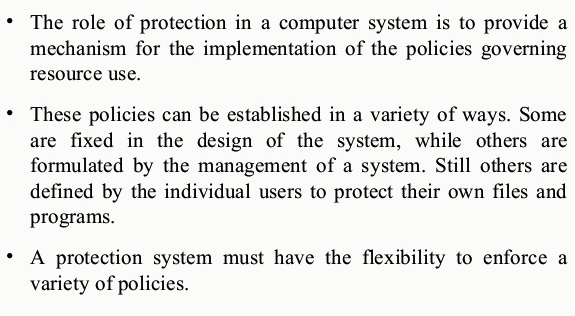


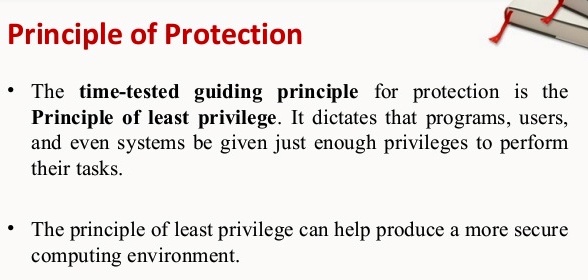
In Figure-2, the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list.  
A drawback of this method is the I/O required for free space list traversal.

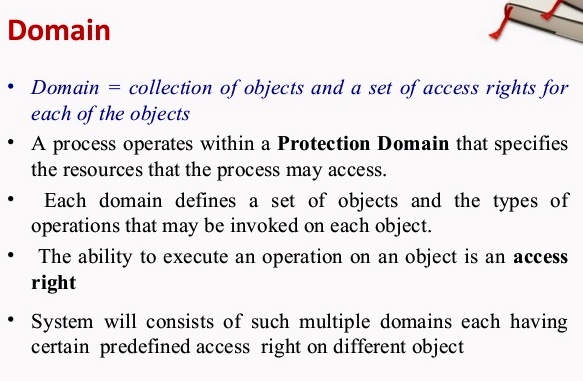
1. **Grouping –**  
   This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks.  
   An **advantage** of this approach is that the addresses of a group of free disk blocks can be found easily.
2. **Counting –**  
   This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block.  
   Every entry in the list would contain:
   * Address of first free disk block
   * A number n

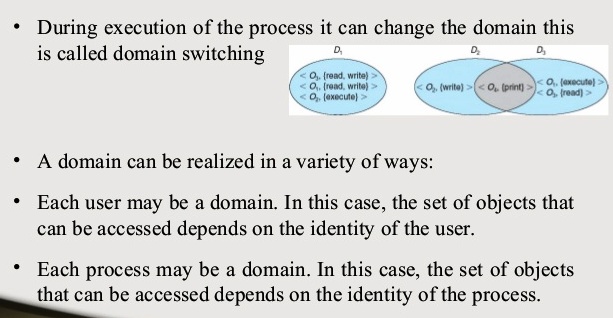
For example, in Figure-1, the first entry of the free space list would be: ([Address of Block 5], 2), because 2 contiguous free blocks follow block 5.





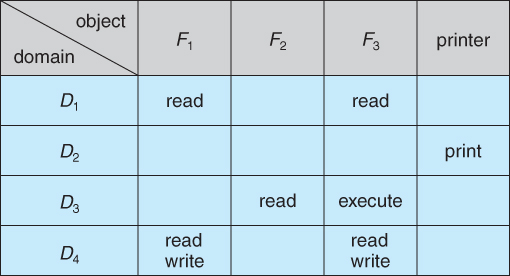




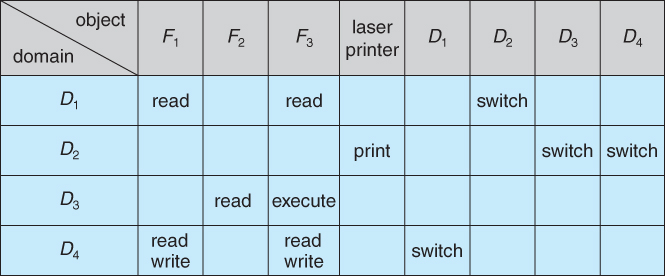


### Access Matrix

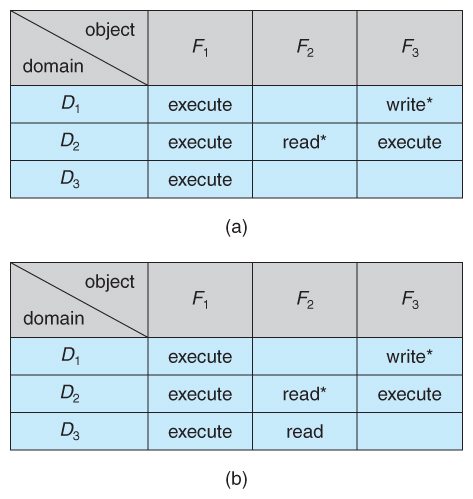
* The model of protection that we have been discussing can be viewed as an ***access matrix,***in which columns represent different system resources and rows represent different protection domains. Entries within the matrix indicate what access that domain has to that resource.

  
**Figure 14.3 - Access matrix.**

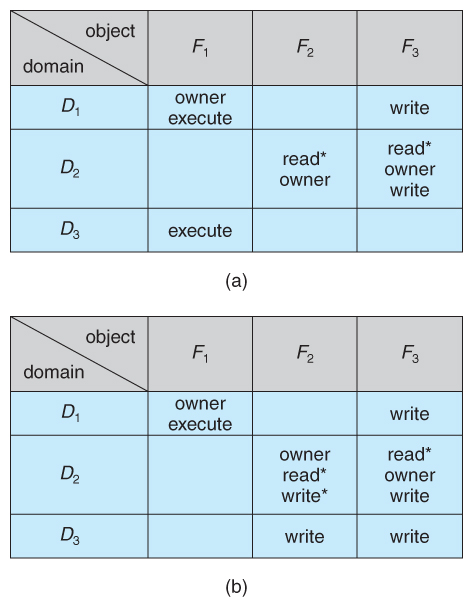
* Domain switching can be easily supported under this model, simply by providing "switch" access to other domains:

  
**Figure 14.4 - Access matrix of Figure 14.3 with domains as objects.**

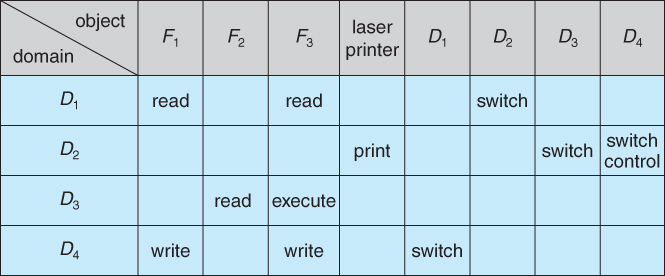
* The ability to ***copy***rights is denoted by an asterisk, indicating that processes in that domain have the right to copy that access within the same column, i.e. for the same object. There are two important variations:
  + If the asterisk is removed from the original access right, then the right is ***transferred,***rather than being copied. This may be termed a ***transfer*** right as opposed to a ***copy*** right.
  + If only the right and not the asterisk is copied, then the access right is added to the new domain, but it may not be propagated further. That is the new domain does not also receive the right to copy the access. This may be termed a ***limited copy*** right, as shown in Figure 14.5 below:

  
**Figure 14.5 - Access matrix with copy rights.**

* The ***owner*** right adds the privilege of adding new rights or removing existing ones:

  
**Figure 14.6 - Access matrix with owner rights.**

* Copy and owner rights only allow the modification of rights within a column. The addition of ***control rights***, which only apply to domain objects, allow a process operating in one domain to affect the rights available in other domains. For example in the table below, a process operating in domain D2 has the right to control any of the rights in domain D4.

  
**Figure 14.7 - Modified access matrix of Figure 14.4**

### 14.5 Implementation of Access Matrix

#### 14.5.1 Global Table

* The simplest approach is one big global table with < domain, object, rights > entries.
* Unfortunately this table is very large ( even if sparse ) and so cannot be kept in memory ( without invoking virtual memory techniques. )
* There is also no good way to specify groupings - If everyone has access to some resource, then it still needs a separate entry for every domain.

#### 14.5.2 Access Lists for Objects

* Each column of the table can be kept as a list of the access rights for that particular object, discarding blank entries.
* For efficiency a separate list of default access rights can also be kept, and checked first.

#### 14.5.3 Capability Lists for Domains

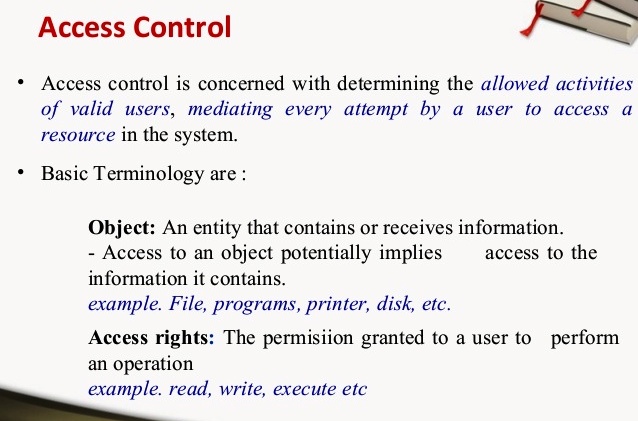
* In a similar fashion, each row of the table can be kept as a list of the capabilities of that domain.
* Capability lists are associated with each domain, but not directly accessible by the domain or any user process.
* Capability lists are themselves protected resources, distinguished from other data in one of two ways:
  + A ***tag,***possibly hardware implemented, distinguishing this special type of data. ( other types may be floats, pointers, booleans, etc. )
  + The address space for a program may be split into multiple segments, at least one of which is inaccessible by the program itself, and used by the operating system for maintaining the process's access right capability list.

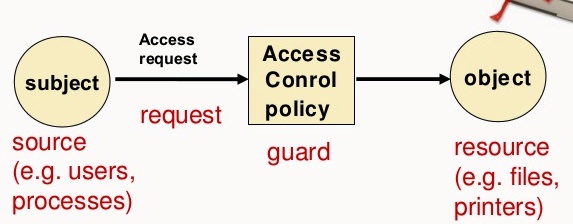
#### 14.5.4 A Lock-Key Mechanism

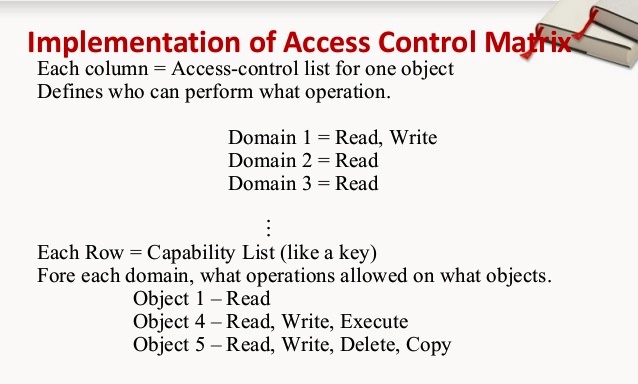
* Each resource has a list of unique bit patterns, termed locks.
* Each domain has its own list of unique bit patterns, termed keys.
* Access is granted if one of the domain's keys fits one of the resource's locks.
* Again, a process is not allowed to modify its own keys.

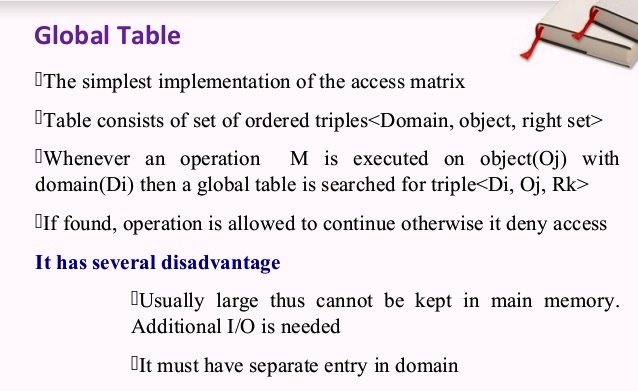
#### 14.5.5 Comparison

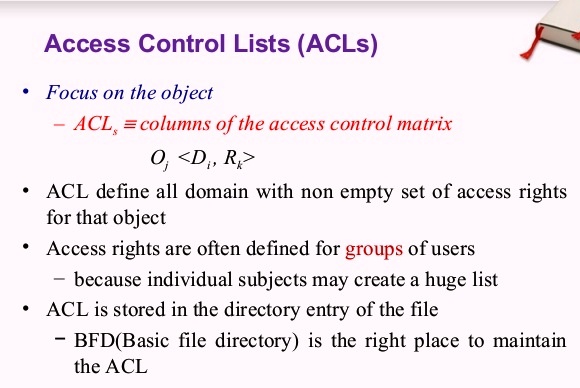
* Each of the methods here has certain advantages or disadvantages, depending on the particular situation and task at hand.
* Many systems employ some combination of the listed methods.

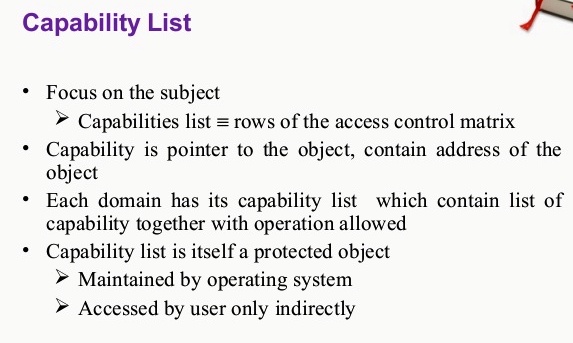


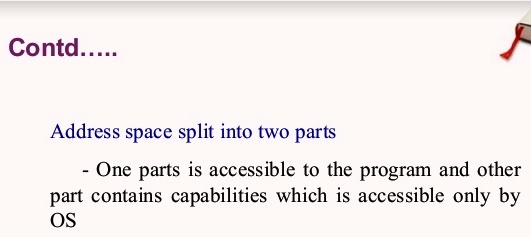


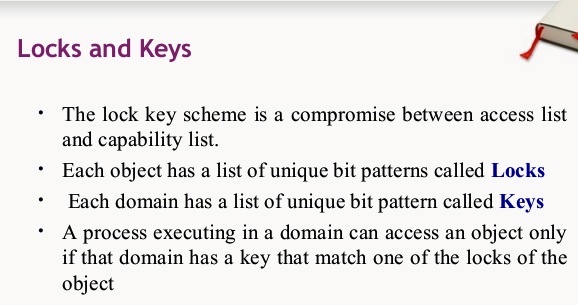


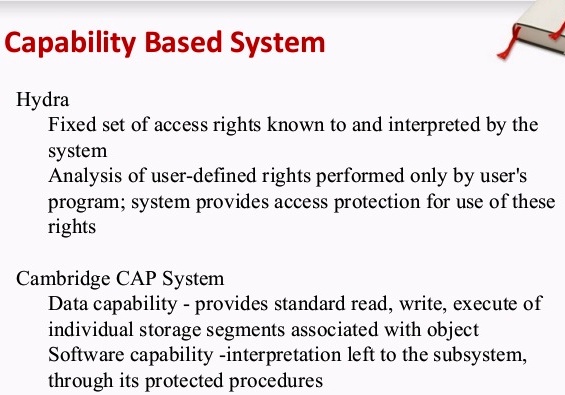


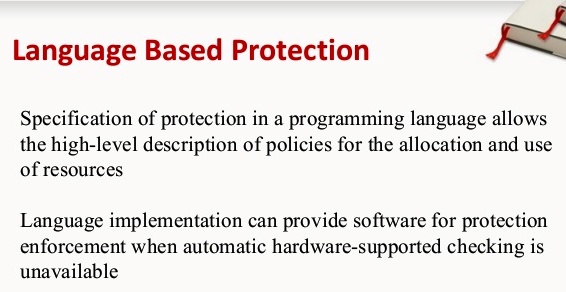












**SECURITY PROBLEMS:**

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. We're going to discuss following topics in this chapter.

* Authentication
* One Time passwords
* Program Threats
* System Threats
* Computer Security Classifications

## Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

* **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
* **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
* **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
* **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

## System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

* **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
* **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
* **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

Types of Network Threats

**Network-delivered threats are typically of two basic types:**

* **Passive Network Threats**: Activities such as wiretapping and idle scans that are designed to intercept traffic traveling through the network.
* **Active Network Threats**: Activities such as Denial of Service (DoS) attacks and SQL injection attacks where the attacker is attempting to execute commands to disrupt the network’s normal operation.

To execute a successful network attack, attackers must typically actively hack a company’s infrastructure to exploit software vulnerabilities that allow them to remotely execute commands on internal operating systems. DoS attacks and shared network hijacking (example: when corporate user is on a public WiFi network) of communications are exceptions.

Attackers typically gain access to internal operating systems via email-delivered network threats which first compromise a set of machines, then install attacker controlled malware, and so provide ability for the attacker to move laterally. This increases the likelihood of not being detected up front while providing an almost effortless entry point for the attacker.

According to a recent Microsoft security intelligence report, **more than 45% of malware requires some form of user interaction**, suggesting that user-targeted email, designed to trick users, is a primary tactic used by attackers to establish their access.