

DEPARTMENT OF COMPUTER APPLICATIONS

MODULE 5

FILE SYSTEM, SYSTEM PROTECTION & SYSTEM SECURITY

FILE CONCEPT

Definition

A file is a named collection of related information stored on secondary storage. It represents data in a persistent form that survives system reboots. Files are managed by the operating system for storage, retrieval, and protection. They provide a logical view of information stored on disk.

Introduction

In an operating system, long-term data storage is achieved using files. Programs and users store data in files so that it can be retrieved later. The operating system abstracts the physical details of storage devices using the file concept. Files allow data to be organized, named, and accessed efficiently. Different types of data such as text, executables, images, and multimedia are stored as files. The OS defines how files are created, accessed, modified, and deleted. Files provide a uniform interface to storage devices. They are stored on disks but accessed logically by users and programs. Without files, managing persistent data would be complex and error-prone. Thus, the file concept is central to modern operating systems.

Explanation

A file consists of two main parts: **data** and **metadata**. The data part stores the actual contents of the file. Metadata contains information such as file name, size, location, permissions, and timestamps.

The operating system defines several **file attributes**, including:

DEPARTMENT OF COMPUTER APPLICATIONS

- Name
- Identifier
- Type
- Size
- Location
- Protection
- Time and date information

Files may be organized logically or physically in different ways. The OS supports operations such as create, open, read, write, seek, close, and delete. Each open file is associated with a file descriptor or handle.

Files may be classified into different types such as:

- Regular files
- Directory files
- Special device files

The OS ensures protection so that only authorized users can access files. File systems may support file sharing and concurrent access. Consistency and reliability are ensured through buffering and caching mechanisms. Files hide the complexity of disk storage from users. They provide a convenient abstraction for persistent data storage.

Example

A text file named notes.txt contains lecture notes. Its metadata includes file size, creation date, and access permissions. A program opens the file, reads the contents, modifies it, and saves it. The OS manages disk access transparently. The user does not need to know where the file is stored physically.

DEPARTMENT OF COMPUTER APPLICATIONS

Conclusion

The file concept provides a logical and persistent way to store information. It simplifies data management and access. Files abstract storage details from users and programs. They are fundamental to operating system functionality. Efficient file management ensures reliable and secure data storage.

ACCESS METHODS

Introduction

Access methods define the manner in which data stored in a file can be read or written by a process. Different applications require different patterns of data access, and the operating system provides suitable mechanisms to support these needs. Access methods determine how records within a file are retrieved, modified, or processed. They help control the movement of the file pointer during read and write operations. Efficient access methods improve file system performance and reduce disk I/O overhead. The operating system hides the complexity of physical disk access by providing logical access methods. These methods ensure that data can be accessed in an organized and predictable manner. Choosing an appropriate access method is essential for efficient file processing. Access methods also contribute to data integrity and consistency. Modern operating systems support multiple access methods to meet diverse application requirements.

ACCESS METHODS: SEQUENTIAL, DIRECT, AND INDEXED ACCESS

Access methods define how data stored in a file can be read or written by a process. The operating system provides different access methods to support various application requirements. These methods control the movement of the file pointer and determine how records within a file are accessed. The most commonly used access methods are sequential access, direct access, and indexed access. Each method is designed to handle specific data retrieval patterns and improve file system efficiency.

DEPARTMENT OF COMPUTER APPLICATIONS

In sequential access, data is accessed in a linear order, starting from the beginning of the file and proceeding record by record until the end. This method is similar to reading data from magnetic tapes. The operating system maintains a file pointer that automatically advances after each read or write operation. Sequential access is simple to implement and requires minimal overhead. It is suitable for applications such as text editors, log processing, and batch systems where data is processed in order. However, accessing a specific record requires reading all preceding records, making it inefficient for random searches.

Direct access, also known as random access, allows a file to be accessed at any location directly. In this method, the file is viewed as a sequence of fixed-size blocks, and the file pointer can jump to any block without reading previous ones. This access method is particularly useful for database systems and large files where specific records need to be retrieved quickly. Direct access provides faster retrieval times for random operations. The operating system calculates the physical location of the requested block using the block number. Although more flexible than sequential access, it requires more complex management by the operating system.

Indexed access combines the advantages of both sequential and direct access methods. In this approach, an index is maintained that contains pointers to different blocks or records within the file. When a file is accessed, the operating system first searches the index to locate the required record and then accesses it directly. Indexed access allows fast searching while still supporting sequential processing of records. It is commonly used in large databases and file systems where quick access to specific records is essential. Maintaining the index introduces additional overhead, but the improved performance justifies the cost.

For example, consider a student records file stored on disk. Using sequential access, each student record is read one by one until the desired record is found. Using direct access, the system jumps directly to the block containing the required student record. Using indexed access, the system consults an index to quickly locate the block containing the student's information. Each access method serves different application needs and performance requirements.

DEPARTMENT OF COMPUTER APPLICATIONS

In conclusion, access methods play a vital role in file management by defining how data is retrieved and stored. Sequential access is simple and efficient for ordered data processing, direct access supports fast random retrieval, and indexed access provides a balance between speed and flexibility. Operating systems support multiple access methods to meet the diverse needs of modern applications. Understanding these methods is essential for efficient file system usage and performance optimization.

Example

Consider a text file that contains student attendance records. When the file is processed sequentially, each record is read one after another until the end of the file. In a database system, a specific student record can be accessed directly using a record number or key value. In indexed access, an index table is used to locate a student's record quickly without scanning the entire file. This reduces access time and improves performance. Each access method serves a different purpose depending on the type of application. The operating system ensures that these methods are implemented efficiently. Thus, applications can choose the most suitable access method for their needs.

Conclusion

Access methods play a vital role in file management by defining how data is accessed within files. They provide flexibility and efficiency in data retrieval and storage. Different access methods support different application requirements. By abstracting disk operations, access methods simplify file usage. Proper selection of an access method improves system performance. Hence, access methods are an essential component of modern file systems.

DIRECTORY AND DISK STRUCTURE

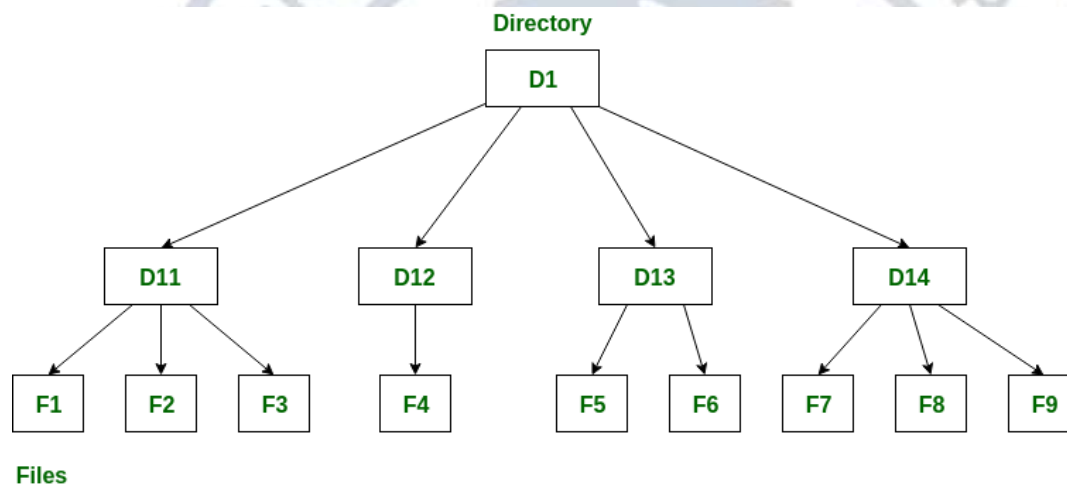
Introduction

Directory and disk structure form the foundation of file organization in an operating system. A directory provides a logical way to group and manage files, while disk structure determines how

DEPARTMENT OF COMPUTER APPLICATIONS

files are physically stored on secondary storage. Directories help users and programs locate files efficiently without knowing their exact physical location. The operating system uses directories to map file names to their corresponding file control blocks. Disk structure defines how data blocks are arranged on the disk surface. Efficient directory and disk structures improve file access time and system performance. They also support file sharing and protection. Modern operating systems use hierarchical directory structures to organize files systematically. Disk structures are designed to minimize seek time and fragmentation. Together, directory and disk structures enable efficient storage, retrieval, and management of data.

Diagram



Explanation

A directory is a special type of file that contains information about other files. It stores entries that map file names to file control blocks, which contain metadata such as file size, location, and access permissions. Directories support operations like creating, deleting, searching, and listing files. Early systems used single-level directories, which were simple but limited. To improve organization, two-level directories were introduced to separate user files. Modern systems use tree-structured directories, which allow hierarchical organization and subdirectories. Disk structure refers to the physical layout of data on the disk, which is divided into tracks, sectors, and blocks. Files are stored as a sequence of disk blocks. The operating system manages free disk

DEPARTMENT OF COMPUTER APPLICATIONS

space using techniques like bitmaps or linked lists. Proper disk structure design reduces fragmentation and improves access speed. Directory and disk structures work together to provide efficient and reliable file storage.

Example

Consider a computer system where a user stores documents in a folder named “Documents” inside their home directory. The directory contains entries for each file, such as “resume.docx” and “notes.txt”. When the user opens a file, the operating system searches the directory to find the corresponding file control block. Using this information, the OS locates the disk blocks where the file data is stored. The disk structure ensures that these blocks are accessed efficiently. This allows the user to retrieve files quickly and reliably. Thus, directories help in logical organization, while disk structure manages physical storage.

Conclusion

Directory and disk structure are essential components of a file system. Directories provide a logical and organized view of files, making file management easier for users. Disk structure ensures efficient physical storage of file data. Together, they improve system performance and usability. Proper directory organization and disk layout reduce access time. Hence, directory and disk structure play a crucial role in modern operating systems.

FILE SYSTEM MOUNTING

Introduction

File system mounting is the process by which an operating system makes a file system available for access. A file system stored on a disk or storage device cannot be used unless it is mounted. Mounting connects a file system to the existing directory structure of the operating system. This allows users and applications to access files transparently. Modern operating systems support multiple file systems simultaneously. Mounting enables the integration of removable and

DEPARTMENT OF COMPUTER APPLICATIONS

permanent storage devices. The operating system verifies the integrity of the file system before mounting it. Each mounted file system is associated with a specific directory called a mount point. File system mounting plays an important role in storage management. It allows flexible and modular use of storage resources.

Explanation

File system mounting involves attaching a file system to a specific point in the directory hierarchy. The mount point is an empty directory where the new file system is attached. Once mounted, the files on the device appear as part of the directory tree. The operating system maintains a mount table that records all mounted file systems. Before mounting, the OS checks the file system type and structure. This ensures that the file system is consistent and usable. Multiple file systems can be mounted at different mount points at the same time. Unmounting a file system safely removes it from the directory structure and ensures that all pending write operations are completed. This prevents data corruption. Mounting allows removable devices such as USB drives and external hard disks to be used efficiently. The entire process is transparent to users and applications.

Example

When a USB drive is connected to a Linux system, the operating system mounts the file system automatically. The USB device may be mounted at a directory such as `/media/usb`. All files stored on the USB drive become accessible through this directory. Users can open, edit, and save files normally. After finishing the work, the user unmounts the device. This ensures that all data is written to the disk safely. The USB drive can then be removed without data loss. This demonstrates how mounting enables access to external storage devices.

Conclusion

File system mounting is essential for accessing stored data. It integrates different file systems into a unified directory structure. Mounting allows flexible use of multiple storage devices. It ensures

DEPARTMENT OF COMPUTER APPLICATIONS

safe and organized access to files. Proper mounting and unmounting prevent data corruption. Thus, file system mounting is a vital function of modern operating systems.

FILE SHARING

Introduction

File sharing allows multiple users or processes to access the same file in a computer system. It is an important feature of multiuser and networked operating systems. File sharing enables collaboration by allowing users to read or modify common data. The operating system controls file sharing through protection and access control mechanisms. Shared files can be accessed locally or over a network. Proper file sharing improves resource utilization and reduces data duplication. However, uncontrolled sharing may lead to data inconsistency and security issues. Therefore, the operating system must carefully manage shared access. File sharing is widely used in distributed systems. It plays a crucial role in modern computing environments.

Explanation

File sharing is implemented by allowing multiple users or processes to access the same file simultaneously or at different times. The operating system uses access permissions to control how files are shared. These permissions specify who can read, write, or execute a file. File sharing may occur within a single system or across a network. In local systems, file sharing is managed through user IDs and group IDs. In networked systems, protocols such as NFS allow files to be shared among multiple machines. The operating system must handle concurrent access carefully to prevent data inconsistency. File locking mechanisms are used to synchronize access when multiple processes attempt to modify a file. Some systems allow read sharing but restrict write sharing. Proper file sharing improves collaboration while maintaining data integrity and security.

DEPARTMENT OF COMPUTER APPLICATIONS

Example

Consider a project file stored on a shared server in a university laboratory. Multiple students can access the file to read project instructions. Only authorized students are allowed to modify the file. The operating system checks access permissions before allowing any operation. If two students attempt to edit the file at the same time, file locking ensures that changes do not conflict. This allows safe and controlled sharing of the file. Thus, file sharing supports collaboration without compromising data integrity.

Conclusion

File sharing enables efficient use of data and system resources. It allows multiple users to access common files easily. The operating system enforces protection mechanisms to ensure secure sharing. Proper synchronization prevents data inconsistency. File sharing is essential in multiuser and networked systems. It enhances collaboration and productivity.

SYSTEM PROTECTION

Introduction

System protection refers to the mechanisms used by an operating system to control access to system resources. It ensures that each user and process can access only those resources for which they are authorized. Protection is necessary in a multiuser and multiprogramming environment where several processes execute simultaneously. Without protection, one process could interfere with the execution of another. System protection helps prevent accidental or malicious misuse of system resources. It maintains the integrity and reliability of the operating system. Protection mechanisms are enforced by both hardware and software. The operating system defines protection policies and ensures they are followed. System protection also supports safe resource sharing. It is a fundamental requirement for secure system operation.

DEPARTMENT OF COMPUTER APPLICATIONS

Explanation

System protection is achieved by defining rules that specify how resources may be accessed. Resources include files, memory, CPU, and I/O devices. The operating system assigns access rights that determine permitted operations such as read, write, or execute. Protection mechanisms prevent unauthorized access and accidental damage. The principle of least privilege is followed, where a process is given only the minimum rights necessary. Protection domains are used to group access rights. A domain specifies the resources a process can access and the operations it can perform. The operating system checks access rights before granting resource usage. Hardware support, such as memory protection and CPU modes, helps enforce protection. Effective protection ensures stable and reliable system operation.

Example

Consider a multiuser system where normal users and administrators coexist. Normal users are allowed to read and modify their own files but cannot access system configuration files. Administrators have additional privileges to modify system resources. When a user attempts to access a protected file, the operating system checks permissions. If access is not allowed, the request is denied. This prevents unauthorized modification of critical system files. Thus, system protection ensures controlled access to resources.

Conclusion

System protection is essential for maintaining system stability and security. It controls how resources are accessed and shared. Protection prevents unauthorized and accidental misuse. The operating system enforces protection policies effectively. Proper protection ensures reliable system operation. It is a core responsibility of modern operating systems.

DEPARTMENT OF COMPUTER APPLICATIONS

PRINCIPLES OF PROTECTION

Introduction

Principles of protection provide guidelines for designing and implementing effective protection mechanisms in an operating system. Protection is necessary to control access to system resources and prevent misuse or unauthorized actions. These principles help ensure that users and processes access only those resources that are required for their legitimate tasks. By following protection principles, operating systems can improve security, reliability, and stability. In multiuser and multiprogramming environments, protection principles become even more important. They help prevent accidental damage and malicious attacks. Protection principles are applied at both hardware and software levels. They guide the design of access control, authentication, and authorization mechanisms. Proper implementation of protection principles reduces system vulnerabilities. Hence, principles of protection form the foundation of secure operating system design.

Explanation

One of the most important principles of protection is the principle of least privilege. According to this principle, a user or process should be granted only the minimum access rights necessary to perform its task. This limits the potential damage caused by errors or malicious actions. Another key principle is fail-safe defaults, which states that access should be denied by default unless explicitly permitted. This reduces the risk of accidental access. The principle of separation of privilege requires that access to critical resources depend on multiple conditions, such as requiring more than one authorization. This increases system security. The principle of economy of mechanism emphasizes simplicity in protection design to reduce errors and vulnerabilities. The principle of complete mediation ensures that every access to a resource is checked for authorization. The principle of open design suggests that security should not depend on secrecy of the system design. Together, these principles guide the creation of robust protection mechanisms in operating systems.

DEPARTMENT OF COMPUTER APPLICATIONS

Example

Consider a multiuser operating system where a normal user runs an application. The application is allowed to access only the user's files and not system configuration files. This follows the principle of least privilege. Access to sensitive system files is denied by default, implementing fail-safe defaults. Administrative tasks require multiple authentication steps, demonstrating separation of privilege. Each time the application tries to access a file, the operating system checks permissions, ensuring complete mediation. These principles together ensure controlled and secure resource access.

Conclusion

Principles of protection provide essential guidelines for secure system design. They help limit unauthorized access and reduce system vulnerabilities. Applying these principles improves system reliability and security. They ensure controlled sharing of resources in multiuser environments. Operating systems rely on these principles to enforce effective protection. Hence, principles of protection are fundamental to secure and stable computing systems.

Domain of Protection and Access Matrix

Introduction:

Domain of protection is a fundamental concept in computer security that defines the set of access rights or privileges a process or user has over system resources. It ensures that operations on files, memory, and devices are performed only by authorized entities. The idea is to prevent unauthorized access and maintain system integrity. Each domain specifies which objects can be accessed and the type of operations allowed, such as read, write, or execute. Protection domains provide a controlled environment for processes. They form the basis for access control mechanisms. Domains can be represented as an abstraction, isolating processes from each other. This separation is crucial to avoid accidental or malicious interference. Operating systems implement domains to enforce security policies. Domains can switch during execution depending on the operation required. The concept is closely associated with the principle of least privilege.

DEPARTMENT OF COMPUTER APPLICATIONS

Proper domain management reduces vulnerabilities. The Access Matrix is a practical method to implement domain-based protection. It maps subjects, objects, and permitted operations in a tabular form.

	Object1	Object2	Object3	...
Domain1	Read	Write	Execute	...
Domain2	Write	Read	None	...
Domain3	None	Execute	Read	...

Explanation:

The Access Matrix is a conceptual model that specifies which domains have which access rights over objects. Rows represent domains (users or processes), and columns represent objects (files, memory segments, devices). Each cell in the matrix lists the access rights that the domain has over the object. For example, Domain1 might have read and write access to Object1 but no access to Object3. The matrix provides a clear and systematic way to enforce security policies. Operating systems use this model to grant or deny access dynamically. Two common implementations of the access matrix are **access control lists (ACLs)** and **capability lists**. ACLs associate each object with a list of domains and their allowed operations, whereas capability lists associate each domain with a list of objects and allowed operations. Domains can change over time depending on the process execution or privilege requirements. The matrix ensures that unauthorized access is not possible, maintaining system integrity. It also allows easy auditing of access rights. The flexibility of the model supports complex security policies. It helps prevent accidental data modification. By controlling access at the domain level, systems reduce the risk of malicious attacks. The access matrix can be sparse, meaning not all cells need to be filled. Security mechanisms often optimize storage and lookup for efficiency. Overall, it is a cornerstone in designing secure operating systems.

Example:

Consider a university system with three types of users: students, faculty, and administrators. Objects include student records, course materials, and administrative files.

- Students (Domain1) can read course materials but cannot modify them.

DEPARTMENT OF COMPUTER APPLICATIONS

- Faculty (Domain2) can read and write course materials and student records.
 - Administrators (Domain3) can read, write, and delete all files.
- The access matrix clearly defines these rights: each domain's permissions for each object are explicitly stated, preventing unauthorized access. When a student attempts to delete a file, the system checks the matrix and denies the operation. Faculty can update grades, but cannot delete administrative files. This ensures secure operation within the system.

Conclusion:

Domain of protection and the access matrix are vital for maintaining security in computer systems. They provide a structured way to control which processes can access which resources. By clearly defining access rights, the system prevents unauthorized access and data breaches. Implementations like ACLs and capability lists make the model practical for real systems. Overall, these mechanisms are foundational for secure and reliable operating systems.

SYSTEM SECURITY

Introduction

System security refers to the protection of computer systems and data from unauthorized access, misuse, modification, and destruction. In modern computing environments, systems are connected through networks and are exposed to various security threats. The operating system plays a central role in enforcing security mechanisms. System security ensures confidentiality, integrity, and availability of data and resources. Confidentiality ensures that data is accessed only by authorized users. Integrity ensures that data is not altered illegally. Availability ensures that system resources remain accessible to legitimate users. Security threats can originate from malicious programs, unauthorized users, or network attacks. As systems become more complex, security challenges increase. Therefore, system security is a critical responsibility of the operating system.

DEPARTMENT OF COMPUTER APPLICATIONS

Explanation

The security problem in an operating system involves protecting system resources against threats. Program threats include malicious software such as viruses, worms, Trojan horses, and logic bombs. Viruses attach themselves to legitimate programs and spread when the program executes. Worms are standalone programs that replicate themselves across systems using networks. Trojan horses appear to be legitimate programs but perform malicious actions secretly. Logic bombs trigger harmful actions when specific conditions are met. System threats target operating system services and resources. Denial-of-service attacks attempt to make a system unavailable to users by overwhelming it with requests. Privilege escalation allows attackers to gain unauthorized administrative access. Network threats arise due to system connectivity. These include eavesdropping, spoofing, man-in-the-middle attacks, and port scanning. To counter these threats, operating systems implement authentication, authorization, encryption, firewalls, and intrusion detection systems. Security policies and regular updates further strengthen system security.

Example

Consider a computer system connected to the internet. A worm spreads through the network by exploiting security vulnerabilities. It consumes system resources and slows down performance. A denial-of-service attack floods a server with requests, preventing legitimate users from accessing services. To prevent such attacks, the operating system uses firewalls to filter traffic and authentication mechanisms to verify users. Antivirus software detects and removes malicious programs. Encryption ensures that data transmitted over the network remains confidential. These security measures protect the system from threats.

Conclusion

System security is essential for protecting data and system resources. It addresses threats from malicious programs, system misuse, and network attacks. The operating system plays a vital role in enforcing security mechanisms. Proper security ensures confidentiality, integrity, and

DEPARTMENT OF COMPUTER APPLICATIONS

availability. Strong security policies and mechanisms prevent unauthorized access. Therefore, system security is a fundamental requirement for reliable and safe computing.

PROGRAM THREATS

Introduction

Program threats are malicious software attacks that exploit vulnerabilities in programs to harm computer systems. These threats are designed to disrupt normal system operations, steal sensitive information, or gain unauthorized access to resources. Program threats usually operate by hiding inside legitimate software or by exploiting programming errors. With the increasing use of computers and networks, program threats have become more sophisticated. They pose serious risks to system security and data integrity. Program threats can spread rapidly and cause significant damage if not detected early. The operating system must identify and control such threats to maintain system stability. Security mechanisms play a vital role in preventing program-based attacks. Understanding program threats is essential for designing secure systems. Hence, program threats are a major concern in system security.

Explanation

Program threats include malicious software that executes unwanted actions on a system. One common program threat is a virus, which attaches itself to a legitimate program and spreads when the program is executed. Viruses can corrupt files, steal data, or disrupt system operations. A worm is another threat that is self-contained and spreads independently through networks without user interaction. Worms consume system and network resources, often causing performance degradation. A Trojan horse appears to be a useful program but performs malicious activities secretly. Unlike viruses, Trojan horses do not replicate themselves. Logic bombs are malicious code segments that activate when specific conditions are met, such as a particular date or event. Buffer overflow attacks occur when a program writes more data than a buffer can hold, allowing attackers to execute arbitrary code. Program threats often exploit poor programming practices and

DEPARTMENT OF COMPUTER APPLICATIONS

lack of input validation. Operating systems counter these threats using access control, memory protection, code signing, and antivirus mechanisms.

Example

Consider a user downloading a free software application from an untrusted source. The program appears legitimate but contains a Trojan horse. Once executed, it secretly installs malicious code that steals user passwords. In another case, a virus infects executable files and spreads when those programs are shared. A worm spreads through a network by exploiting security vulnerabilities in running services. These attacks compromise system security and performance. Antivirus software detects and removes malicious programs. Operating system security features help prevent such attacks.

Conclusion

Program threats pose a serious risk to system security and data integrity. They exploit vulnerabilities in software to perform malicious actions. Viruses, worms, Trojan horses, and logic bombs are common examples. Effective operating system security mechanisms are required to detect and prevent these threats. Secure programming practices reduce vulnerabilities. Understanding program threats is essential for building secure and reliable systems.

SYSTEM THREATS

Introduction

System threats are attacks that target the operating system and its resources rather than individual application programs. These threats aim to disrupt normal system functioning, gain unauthorized control, or deny services to legitimate users. System threats exploit weaknesses in operating system design, configuration, or implementation. As modern operating systems support multitasking, networking, and user interaction, they become more vulnerable to such attacks. System threats can affect system availability, integrity, and reliability. They often result in system

DEPARTMENT OF COMPUTER APPLICATIONS

crashes, data loss, or performance degradation. The operating system must detect and prevent these threats to ensure stable operation. System threats are more dangerous than simple program threats because they can compromise the entire system. Therefore, understanding system threats is essential for ensuring system security.

Explanation

System threats directly target operating system services and resources. One major system threat is the denial-of-service attack, where the attacker attempts to make the system unavailable to legitimate users. This is achieved by flooding the system with excessive requests or exploiting resource exhaustion. Another serious system threat is privilege escalation, in which an attacker gains higher access rights than permitted. This allows the attacker to perform administrative operations illegally. Resource misuse is another threat where system resources such as CPU time, memory, or disk space are consumed excessively, degrading performance. System threats may also involve modifying system files or configurations. Race conditions and improper synchronization can be exploited to bypass security checks. Weak authentication mechanisms further increase vulnerability. To counter system threats, operating systems implement access control, auditing, authentication, and resource management techniques. Kernel protection and secure system design are essential to prevent system-level attacks.

Example

Consider a server that receives an overwhelming number of requests from an attacker. The server becomes overloaded and cannot respond to legitimate users, resulting in a denial-of-service condition. In another case, a normal user exploits a system bug to gain administrator privileges. This allows the user to modify system files and disrupt services. These attacks compromise system availability and integrity. Security patches and strict access control help prevent such threats. Monitoring system activity also helps in early detection.

DEPARTMENT OF COMPUTER APPLICATIONS

Conclusion

System threats pose a serious risk to operating system stability and security. They target core system services and resources. Denial-of-service attacks and privilege escalation are common system threats. Effective operating system protection mechanisms are essential to prevent such attacks. Secure system design and monitoring reduce vulnerabilities. Hence, addressing system threats is crucial for reliable system operation.

