OPERATING SYSTEMS

Input - Output Management and Security - 5

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Implementation of Access Matrix, Access Control, Access Rights

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 5

10 Hours

Unit-5:Unit 5: IO Management and Security

I/O Hardware, polling and interrupts, DMA, Kernel I/O Subsystem and Transforming I/O Requests to Hardware Operations - Device interaction, device driver, buffering System Protection: Goals, Principles and Domain of Protection, Access Matrix, Access control, Access rights. System Security: The Security Problem,Program Threats, System Threats and Network Threats. Case Study: Windows 7/Windows 10

OPERATING SYSTEMS

Course Outline

47	I/O Hardware, polling and interrupts	13.1,13.2
48	DMA	13.2.3
49	Transforming I/O Requests to Hardware Operations, Device interaction, device driver, buffering.	13.5
50	Goals, Principles and Domain of Protection	14.1-14.3
51	Access Matrix	14.4
52	Access control, Access rights	14.5-14.7
53	The Security Problem	15.1
54	Program Threats	15.2
55	System Threats and Network Threats	15.3
56	Case Study : Windows File System	17.5

- Implementation of Access Matrix
- Access Control
- Access Rights

Implementation of Access Matrix

- In general, the matrix will be sparse; that is, most of the entries will be empty.

- Although data structure techniques are available for representing sparse matrices, they are not particularly useful for this application, because of the way in which the protection facility is used.

Implementation of Access Matrix

- **Global Table:**
  - The simplest implementation of the access matrix is a global table consisting of a set of ordered triples < domain, object, rights-set >.
  - Whenever an operation M is executed on an object Oj within domain Di , the global table is searched for a triple <Di , Oj , Rk >, with M ∈ Rk .
  - If this triple is found, the operation is allowed to continue; otherwise, an exception (or error) condition is raised.

Implementation of Access Matrix

- **Global Table:**
  - This implementation suffers from several drawbacks. The table is usually large and thus cannot be kept in main memory, so additional I/O is needed.
  - Virtual memory techniques are often used for managing this table.
  - In addition, it is difficult to take advantage of special groupings of objects or domains.

Implementation of Access Matrix

- **Access Lists for Objects:**
  - List for each object consists of ordered pairs <domain, rights-set >, which define all domains with a nonempty set of access rights for that object.
  - This approach can be extended easily to define a list plus a default set of access rights.
  - When an operation M on an object Oj is attempted in domain Di , we search the access list for object Oj , looking for an entry <Di , Rk > with $M \in Rk$.
  - If the entry is found, we allow the operation; if it is not, we check the default set.
  - If M is in the default set, we allow the access. Otherwise, access is denied, and an exception condition occurs.

Implementation of Access Matrix

- **Capability Lists for Domains:**
  - A capability list for a domain is a list of objects together with the operations allowed on those objects.
  - An object is often represented by its physical name or address, called a capability.
  - To execute operation M on object Oj , the process executes the operation M, specifying the capability (or pointer) for object Oj as a parameter.
  - Simple possession of the capability means that access is allowed.
  - The capability list is associated with a domain, but it is never directly accessible to a process executing in that domain

Access Control

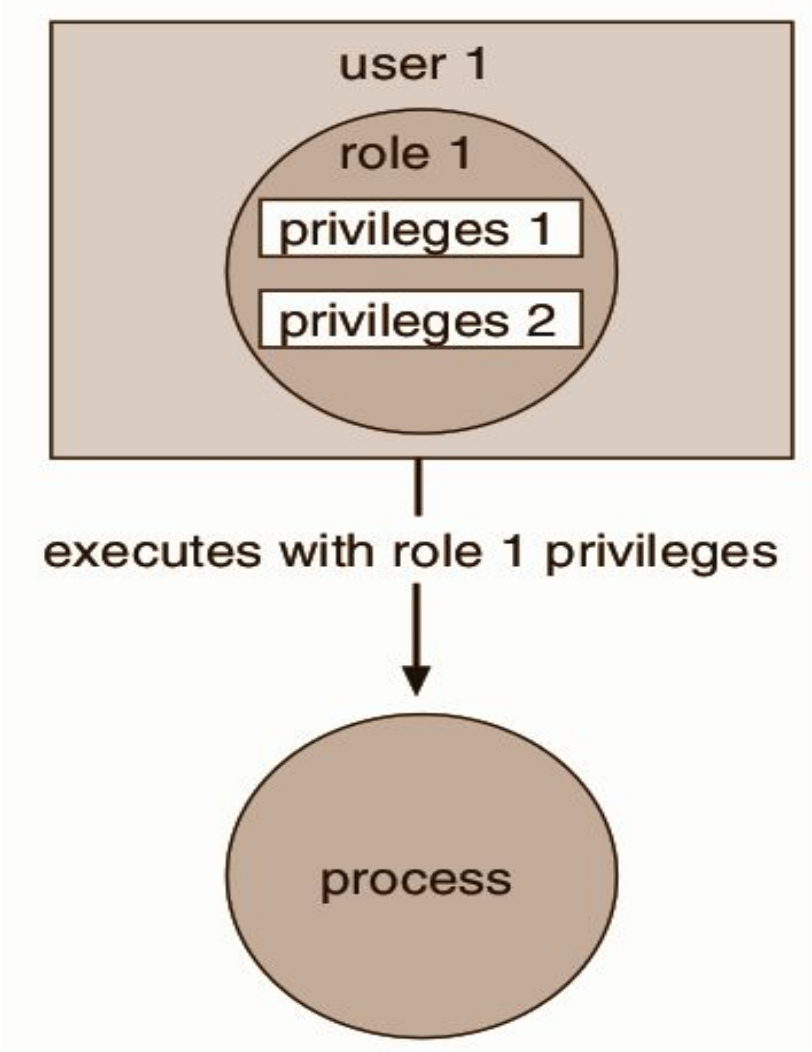
- Each file and directory is assigned an owner, a group, or possibly a list of users, and for each of those entities, access-control information is assigned.
- A similar function can be added to other aspects of a computer system.
- Solaris 10 advances the protection available in the operating system by explicitly adding the principle of least privilege via role-based access control (RBAC).
- This facility revolves around privileges.

Access Control

- A **Privilege** is the right to execute a system call or to use an option within that system call (such as opening a file with write access).
- Privileges can be assigned to processes, limiting them to exactly the access they need to perform their work.
- Privileges and programs can also be assigned to roles.
- Users are assigned roles or can take roles based on passwords to the roles.

Access Control

- Notice that this facility is similar to the access matrix



Role-based access control in Solaris 10.

Access Rights

- In a dynamic protection system, we may sometimes need to revoke access rights to objects shared by different users

- Various questions about revocation may arise:
  - **Immediate versus delayed:** Does revocation occur immediately, or is it delayed ? If revocation is delayed, can we find out when it will take place ?
  - **Selective versus general:** When an access right to an object is revoked, does it affect all the users who have an access right to that object, or can we specify a select group of users whose access rights should be revoked ?
  - **Partial versus total:** Can a subset of the rights associated with an object be revoked, or must we revoke all access rights for this object ?
  - **Temporary versus permanent:** Can access be revoked permanently, that is, the revoked access right will never again be available, or can access be revoked and later be obtained again ?

Access Rights

- Schemes that implement revocation for capabilities include the following
  - **Reacquisition:**
    - Periodically, capabilities are deleted from each domain. If a process wants to use a capability, it may find that that capability has been deleted.
    - The process may then try to reacquire the capability. If access has been revoked, the process will not be able to reacquire the capability.
  - **Back-pointers:**
    - A list of pointers is maintained with each object, pointing to all capabilities associated with that object.
    - When revocation is required, we can follow these pointers, changing the capabilities as necessary.
    - This scheme was adopted in the MULTICS system.
    - It is quite general, but its implementation is costly.

Access Rights

- Schemes that implement revocation for capabilities include the following
  - **Indirection:**
    - The capabilities point indirectly, not directly, to the objects.
    - Each capability points to a unique entry in a global table, which in turn points to the object.
    - We implement revocation by searching the global table for the desired entry and deleting it.
    - Then, when an access is attempted, the capability is found to point to an illegal table entry.
    - Table entries can be reused for other capabilities without difficulty, since both the capability and the table entry contain the unique name of the object.
    - The object for a capability and its table entry must match.

Access Rights

- Schemes that implement revocation for capabilities include the following
  - **Keys:**
    - A key is a unique bit pattern that can be associated with a capability.
    - This key is defined when the capability is created, and it can be neither modified nor inspected by the process that owns the capability.
    - A master key is associated with each object; it can be defined or replaced with the set-key operation.
    - When a capability is created, the current value of the master key is associated with the capability.
    - When the capability is exercised, its key is compared with the master key.
    - If the keys match, the operation is allowed to continue; otherwise, an exception condition is raised.
    - Revocation replaces the master key with a new value via the set-key operation, invalidating all previous capabilities for this object.

Access Rights

- This scheme does not allow selective revocation, since only one master key is associated with each object.
- If we associate a list of keys with each object, then selective revocation can be implemented.
- Finally, we can group all keys into one global table of keys.
- A capability is valid only if its key matches some key in the global table.
- We implement revocation by removing the matching key from the table.
- With this scheme, a key can be associated with several objects, and several keys can be associated with each object, providing maximum flexibility.
- In key-based schemes, the operations of defining keys, inserting them into lists, and deleting them from lists should not be available to all users.

Access Rights

- In particular, it would be reasonable to allow only the owner of an object to set the keys for that object.

- This choice, however, is a policy decision that the protection system can implement but should not define.

Topics Uncovered in this Session

- Implementation of Access Matrix
- Access Control
- Access Rights

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com