Access Control Matrix

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Comparison of Implementations

- **Goals of Protection**
- **Principles of Protection**
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Comparison of Implementations

Protection

Mechanisms and policy to keep programs and users from accessing or changing stuff they should not do

Goals of Protection

- Operating system consists of a collection of objects (hardware or software)
- Each object has a unique name and can be accessed through a well-defined set of operations
- Protection problem – to ensure that each object is accessed correctly and only by those processes that are allowed to do so

Principles of Protection

- Principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks
- Separate policy from mechanism
  - Mechanism : the stuff built into the OS to make protection work
  - Policy : the data that says who can do what to whom

- Goals of Protection
- Principles of Protection
- **Domain of Protection**
- Access Matrix
- Implementation of Access Matrix
- Comparison of Implementations

Domain of protection

- A computer system can be seen as a collection of **Objects** and **Processes**
- Objects may be **software** (files, programs, semaphores) or **Hardware** (CPU, RAM, printer)
- A process should only be allowed to access resources for which it has authorisation and follow the **Need to know** principle

Need to know principle

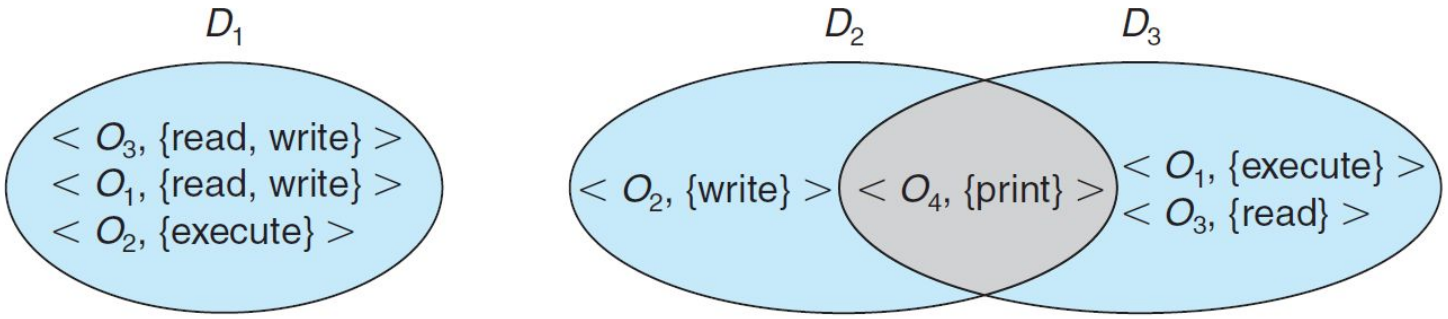
- It states that a process should only have access to:
  - Those objects it needs to accomplish its task
  - Only in the modes for which it needs access
  - Only during the time frame when it needs access

Domain Structure

- To ensure that the scheme is followed, processes run within a **Protection Domain**
- The **Protection Domain** specifies the set of objects and types of operation that may be invoked on each object
- The ability to execute an operation is called **access right**
- Access rights are defined as an ordered pair <object-name, rights-set>

Domain Space

- A domain is a collection of access rights
- Given below is an example domain space



Domain Switching

- The association between a process and a domain may be static or dynamic
  - If the association is static, then the need-to-know principle requires a way of changing the contents of the domain dynamically
  - If the association is dynamic, then there needs to be a mechanism for **domain switching**
- Domain switching simply means the ability of a process to switch from one domain to another

Domain

- Domain may be realised in three ways:
  - Each **user** may be a domain. Domain switching occurs when a user logs out and another one logs in
  - Each **Process** may be a domain. Domain switching occurs when one process sends a message to another and waits for response
  - Each **Procedure** may be a domain. Domain switching occurs when a procedure call is made

- Goals of Protection
- Principles of Protection
- Domain of Protection
- **Access Matrix**
- Implementation of Access Matrix
- Comparison of Implementations

Elements of the Access Matrix

- Rows - Domains
- Columns - Objects
- Entry(i,j) - Set of operations that a process executing in domain Di can invoke on object Oj

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Role of the User

- Ensure that a process executing in domain Di can access only those objects specified in row i, and then only as allowed by the access-matrix entries
- Decide the domain in which each process executes
- Decide the contents of the access-matrix entries

Switch

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Copy Right

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Variants of Copy Right

- A right is copied from access(i, j) to access(k, j); it is then removed from access(i, j)
- Propagation of the copy right may be limited. That is, when the right R∗ is copied from access(i, j) to access(k, j), only the right R (not R∗) is created. A process executing in domain Dk cannot further copy the right R

Owner

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Control

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- **Implementation of Access Matrix**
- Comparison of Implementations

How can the access matrix be implemented?

- It is a sparse matrix, but sparse handling data structures cannot be used
- Reason: Protection facility
- Implementation methods:
  - Global Table
  - Access Lists for Objects
  - Capability lists for Domains
  - A lock-key mechanism

Global Table

1. Consist of Ordered triplets <*domain, object, right-set*>
2. Right-set = [right,write,read-write,execute]
3. Search <Di,Oi,M> (M belongs to any right-set)
4. If the triplet is found the operation is allowed to execute, else not
5. **Drawbacks**
    a. Table is large, hence additional memory needed extra I/O
    b. Difficult to take advantage of special grouping of domain or object
        i. Eg., If everyone has right to read, object should have read in every domain

Access List

1. Creation of access list for every object, hence empty spaces can be removed by defining only non empty entries.
2. Resulting list for each object consist of <*domain,right-set*> pair
3. *Default set (*an access list) consist of allowable operations.
4. Operation M, on object Oi, Domain Di is allowed if the access list of Oi contains <Di,M> pair or if M is present in *default set.*
5. Else Operations is not allowed.

Example

Alice	R/W	R	R	-
Bob	R	R/W		R
Carol	R	R	R/W	R
Dave		R/W	R/W	R
	aaa	**bbb**	**ccc**	**ddd**

aaa -- Alice:R/W, Bob:R, Carol:R

bbb -- Bob:R/W, Dave:R/W, *Others*:R

ccc -- Alice:R, Carol:R/W, Dave:R/W

ddd -- Bob:R, Carol:R, Dave:R

Capability List

1. Rather than grouping columns as access list we group each rows with its domain. It is called as **Capability list**
2. Hence, Capability list for a domain is a list of objects together with the operations allowed
3. **Capability:** Object is represented by its physical name or address
4. Execution of operation M on Object O takes place using capability
5. Hence Capabilities should be secure

Example

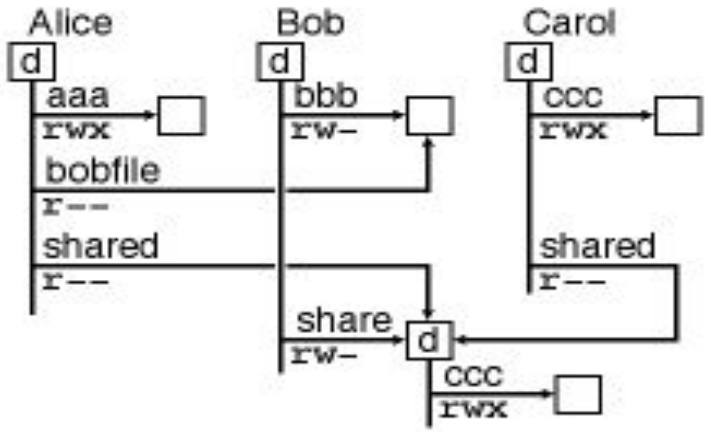
Alice	R/W	R	R	-
Bob	R	R/W		R
Carol	R	R	R/W	R
Dave		R/W	R/W	R
	aaa	**bbb**	**ccc**	**ddd**

Alice -- aaa:R/W, bbb:R, ccc:R

Bob -- aaa:R, bbb:R/W, ddd:R

Carol -- aaa:R, bbb:R, ccc:R/W, ddd:R

Dave -- bbb:R/W, ccc:R/W, ddd:R



Inherent Protection

- Inherently protected pointers provides protection to application level for capability
- Capability are distinguished from other data at application level as:
  - Object Tag (One bit representation), It is implemented by hardware
  - Address space associated with program can be split into two:
    - Accessible by program - Contains normal data, instructions
    - Accessible by OS - Contains capabilities
- Example : Hydra, Cambridge cap system for Capability list domain

Lock-Key Mechanism

- The lock–key scheme is a compromise between access lists and capability lists
- Each object has a list of unique bit patterns, called locks
- Similarly, each domain has a list of unique bit patterns, called keys
- A process executing in a domain can access an object only if that domain has a key that matches one of the locks of the object
- As with capability lists, the list of keys for a domain must be managed by the operating system on behalf of the domain
- Users are not allowed to examine or modify the list of keys (or locks) directly

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- **Comparison of Implementations**

Global Table

- The table can be quite large and often cannot take advantage of special groupings of objects or domains
- Usage is simple

Access Lists

- Access lists correspond directly to the needs of users
- When a user creates an object, he can specify which domains can access the object, as well as what operations are allowed
- Access-right information for a particular domain is not localized, determining the set of access rights for each domain is difficult
- Every access to the object must be checked, requiring a search of the access list
- In a large system with long access lists, this search can be time consuming

Capability Lists

- Capability lists do not correspond directly to the needs of users,but they are useful for localizing information for a given process
- The process attempting access must present a capability for that access
- Then, the protection system needs only to verify that the capability is valid

Modern Usage

- Most systems use a combination of access lists and capabilities
- When a process first tries to access an object, the access list is searched
- If access is denied, an exception condition occurs. Otherwise, a capability is created and attached to the process
- Additional references use the capability to demonstrate swiftly that access is allowed. After the last access, the capability is destroyed

Modern Usage

- Consider a file system in which each file has an associated access list
- When a process opens a file, the directory structure is searched to find the file, access permission is checked, and buffers are allocated
- All this information is recorded in a new entry in a file table associated with the process
- The operation returns an index into this table for the newly opened file
- All operations on the file are made by specification of the index into the file table
- The entry in the file table then points to the file and its buffers

Modern Usage

- When the file is closed, the file-table entry is deleted
- Since the file table is maintained by the operating system, the user cannot accidentally corrupt it
- User can access only those files that have been opened
- Right to access must still be checked on each access, and the file-table
- Entry has a capability only for the allowed operations
- If a file is opened for reading, then a capability for read access is placed in the file-table entry

References

- Abraham Silberschatz, Greg Gagne and Peter Baer Galvin. (2014) **Operating System Concepts**

Thank You