

Overview

- Access control plays a principal role in a security policy and in computer security in general
- Assuming a user has been authenticated, access control defines the resources to which the authenticated user is to be granted access along with type of access to be granted
- Material drawn from:
 - *Introduction to Computer Security* by Bishop
 - *Computer Security: Principles and Practice* by Stallings
 - *Access Control: Principles and Practice* by Sandhu and Samarati



Access Control

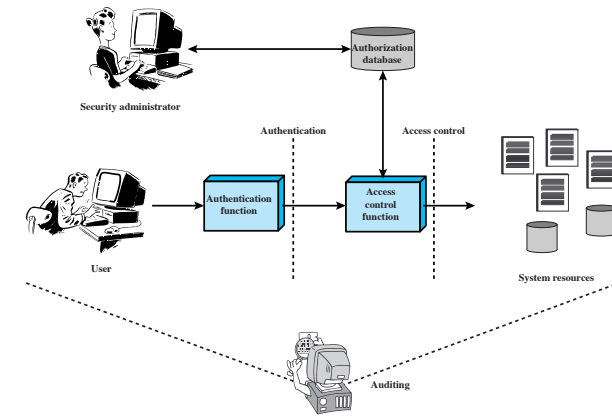


Figure 4.1 Relationship Among Access Control and Other Security Functions (based on [SAND94])



Protection State

- A security policy will partition states into secure and not secure
- The state of a system is the collection of all current values of all of the components of the system
- The subset of these values and components that deal with the protection of the system is the protection state
- Changes in the system can therefore result in changes in the protection state
- These changes must be constrained: our access control model must define a set of authorised states and a set of allowed operations on those states



Access Control Matrix

- An allowed operation on an authorised state yields another authorised state
- The access control matrix is the most precise model used to describe the protection state
- It characterises the rights of each subject with respect to every entity in the system
- The set of objects **O** is the set of all protected entities and each is used typically to contain or receive information
- The set of subjects **S** is the set of all entities that are capable of accessing objects (e.g. processes, users)



Access Control Matrix

- A subject is held accountable for their actions and an audit trail may be used to record the association of a subject with security-relevant actions
- The access control matrix **A** captures the relationships between the subjects and the objects in terms of rights, drawn from a set **R**
- The set of rights **R** depends on the context and object begin protected e.g. a user might be given permission to credit a bank account but not to debit it even though both operations change the underlying balance
- The owner of an object may have permission to grant or revoke access to it thereby making modifications to the site security policy

Access Control Matrix

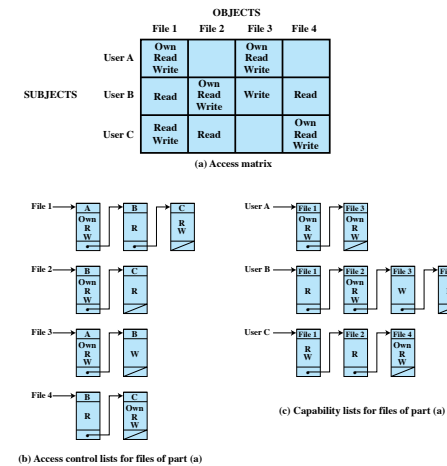


Figure 4.3 Example of Access Control Structures

Access Control Lists

- In practice the access control matrix is too sparse to be implemented directly and is usually decomposes in one of two ways
- The matrix may be decomposed according to columns, yielding **access control lists** or ACLs
- For each object the ACL lists users and their permitted access rights
- The ACL may contain a default entry that affords users not explicitly listed in the ACL least privilege e.g. read access
- ACLs mean that the problems associated coarse grained owner, group and world **abbreviations** common in access control schemes can be alleviated

Capability Tickets

- The ACL data structure is not convenient for analysing access rights by user
- Decomposition by row yields **capability tickets**
- A ticket specifies authorised objects and rights for a particular user
- It is easy to determine the access rights per user but it is more difficult to determine which users have access to any given resource
- An authorisation table is a non-sparse data structure that combines ACLs and capability tickets where each row captures one access right of one subject to one resource
- Sorting by object yields an ACL, sorting by subject a capability list

Authorisation Table

Table 4.1 Authorization Table for Files in Figure 4.3

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Navigation icons: back, forward, search, etc.

Protection State Transitions

- Note that a subject can itself also be an object e.g. a process (acting as a subject) may have the right to signal another process (acting as an object)
- Let the initial state be $X_0 = (S_0, O_0, A_0)$
- Let the set of state transitions be represented by the set of operations τ_1, τ_2, \dots
- The successive states are X_1, X_2, \dots
- $X_i \vdash_{\tau_{i+1}} X_{i+1}$ represents the transition from state X_i to state X_{i+1} by operation τ_{i+1}
- $X \vdash^* Y$ represents the transition from state X to Y over a series of operations

Navigation icons: back, forward, search, etc.

Primitive Transition Operations

- The following primitive operations can be carried out on the access control matrix:
 1. Create subject S adds a row and column for S (S must not already exist as a subject or object and no rights are assigned)
 2. Create object O adds a column for O (O must not already exist as a subject or object and no rights are assigned)
 3. Enter *right* into $A[S, O]$
 4. Delete *right* from $A[S, O]$
 5. Destroy subject S
 6. Destroy object O

Navigation icons: back, forward, search, etc.

Combining Transition Operations

- Primitive operations can be combined to build commands:


```
command spawn_process(P, C) {
    Create subject C;
    Enter own into A[P, C];
    Enter read into A[P, C];
    Enter write into A[P, C];
    Enter read into A[C, P];
    Enter write into A[C, P];
}
```
- We put `read` and `write` into the parent and child so that they can signal one another

Navigation icons: back, forward, search, etc.

Conditional Commands

- The `if` command allows a primitive operation to proceed only if some predefined condition is valid and can be used in conjunction with `and`, `or`, `not` and `in`:

```
command grant_read_file(P, F, Q) {  
    if own in A[P, F];  
    then  
        Enter read into A[Q, F];  
    }  
}
```

- The above will prevent users who do not own a file granting read access to it to others



Discretionary Access Control

- So far a user, the owner of an object, has had the ability to grant and revoke its access rights for other users
- Access control is at the **discretion** of the object's owner and subjects exercise some control over security policy
- **Discretionary access control** can pose a security risk since an attacker may be able to trick an unsuspecting user into disclosing information
- For example a user might execute a Trojan that modifies access rights to some sensitive data or sends it to the network, a printer etc.



Mandatory Access Control

- Can we have a security policy that cannot be modified by the subjects (unless given explicit permission to do so) and where access to objects must be **mandated** by the policy?
- Under **mandatory access control** the policy controls access to an object and an individual user cannot alter that access
- Mandatory access control is enforced by the mechanisms that implement the security policy



Role Based Access Control

- Under DAC access rights are defined according to the identity of the users and the groups to which they belong
- RBAC is based on the roles users assume in the system rather than on their identity and access rights are assigned to roles rather than individual users
- Users are assigned roles according to their responsibilities
- The relationship of users to roles is many to many as is the relationship of roles to resources
- Each role has specific access rights to one or more resources and because a user is assigned the role that enables them to perform only what they require, the principle of least privilege is enforced



Role Based Access Control

- Various RBAC models have been proposed and can incorporate role hierarchies with a role automatically inheriting the access rights of the roles it dominates
- Constraints can be added e.g. mutually exclusive roles require a user takes on only a single role during a session and an access right can only appear in a single role
- The following example is from a real bank where a hierarchy exists and rights are inherited within the same function
- When a user invokes an application it grants access based on a centrally held security policy that defines that user's role and the rights associated with that role



Role Based Access Control

Table 4.4 Functions and Roles for Banking Example

(a) Functions and Official Positions

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
...
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

(b) Permission Assignments

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
...

(c) PA with Inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
...



Terminology Recap

- A **security policy** partitions the system into into a set of authorised or secure states and a set of unauthorised or insecure states
- A **secure system** is one that starts in an authorised state and cannot enter an unauthorised state
- A **breach of security** occurs when a system enters an unauthorised state and we monitor by **auditing**
- A **security model** expresses a policy in terms of operations and transitions that can be reasoned about and implemented by security mechanisms
- **Security mechanisms** enforce the security policy



Confidentiality Policy

- A security policy addresses each of confidentiality, integrity and availability
- The confidentiality policy addresses the confidentiality requirements of the security policy
- A **military security policy** is one aimed at providing confidentiality
- Such policies are required where a breach of confidentiality would be catastrophic e.g. date and location of the invasion
- Integrity and availability are important but maintaining confidentiality is paramount



Integrity Policy

- The integrity policy addresses the integrity requirements of the security policy
- A **commercial security policy** is one aimed at providing integrity
- Commercial firms need to prevent tampering with their data
- For example if the confidentiality of a bank's computer is compromised a customer's account balance or personal details may be revealed
- Such breaches may embarrass the bank but the threat to the bank's overall business is minimal
- If the integrity of the bank's computer was compromised and customer balances could be modified a real threat to the bank's existence would arise