Computer Security: Principles and Practice

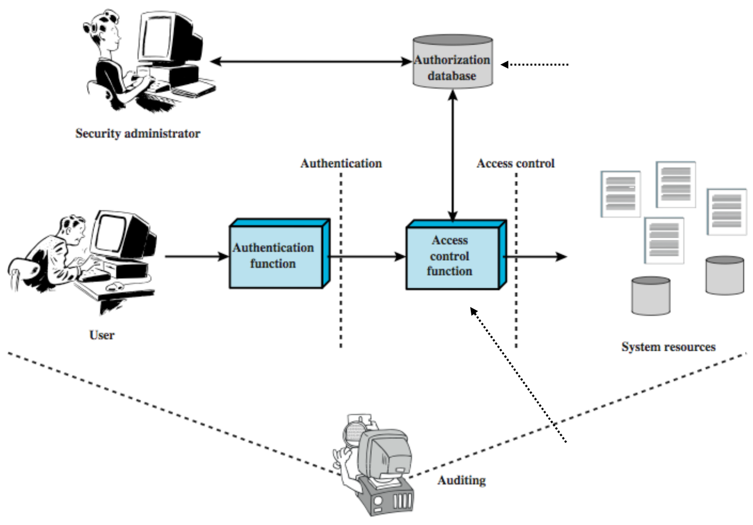
Chapter 4 – Access Control

First Edition
by William Stallings and Lawrie Brown

Access Control

- “The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner“
- central element of computer security
- an access control *policy*
  - defines who (e.g., user, user group, process, etc.) can access what (system resources such as files, channels, services, etc.), in which manner (e.g., read, write, execute, etc.), and under what circumstences (e.g., time, location, history, etc.)
- access control *enforcement*
  - system components that ensure that the system operates in compliance with the access control policy
  - it should be impossible to circumvent the access control enforcement function
  - examples: firewalls, OS reference monitor

General Model of Access Control



Types of Access Control Policies

- **Discretionary access control (DAC)**
  - based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do with the protected resources
- **Mandatory access control (MAC)**
  - based on comparing security labels (which indicate how sensitive or critical system resources are) with security clearances (which indicate system entities are eligible to access certain resources)
- **Role-based access control (RBAC)**
  - based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles

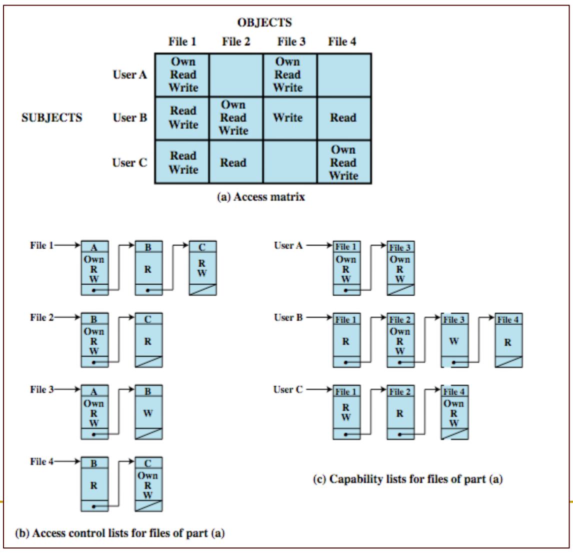
Basic Elements of AC

- **subject** - entity that can access objects
  - e.g., a user, user group, or a process representing a user
- **object** - access controlled resource
  - e.g. files, directories, records, programs, etc.
- **access right** - way in which subject accesses an object
  - e.g. read, write, execute, delete, create, search

Discretionary Access Control

- often represented in terms of an ***access matrix***
  - lists subjects in one dimension (rows)
  - lists objects in the other dimension (columns)
  - each entry specifies access rights of the specified subject to that object
- the access matrix is often sparse
- can be decomposed by either row (credentials) or column (access control lists)
- another efficient representation is the authorization table, which contains (subject, object, access right) triplets
  - can be sorted (indexed) either by subject (→ credentials) or by objects (→ ACLs)

Decomposition of an AC Matrix



OBJECTS

SUBJECTS		File 1	File 2	File 3	File 4
	User A	Own Read Write		Own Read Write	
	User B	Read Write	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

File 1 → A Own R W → B R → C R W
File 2 → B Own R W → C R
File 3 → A Own R W → B W
File 4 → B R → C Own R W
User A → File 1 Own R W → File 3 Own R W
User B → File 1 R → File 2 Own R W → File 3 W → File 4 R
User C → File 1 R W → File 2 R → File 4 Own R W

(b) Access control lists for files of part (a)

(c) Capability lists for files of part (a)

A more general DAC model
(Lampson, Graham, Denning)

- subjects are allowed to alter the protection state (represented by the AC matrix)
  - copy flag – transfer of the given access right to another subject (w/o copy flag)
  - owner – can grant any access right on the given object
  - control – can delete access rights assigned to the given subject

OBJECTS

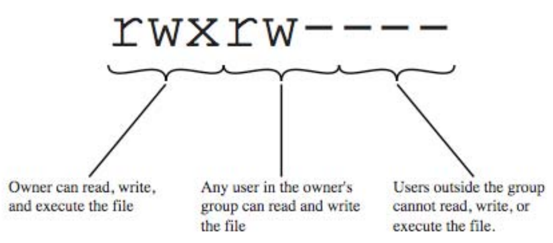
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_1	P_1	P_2	D_1	D_2
	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
SUBJECTS	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Allowed commands for S_0

- transfer a/a* to (S, X)
  - precondition: (S_0, X) contains a*
- grant a/a* to (S, X)
  - precondition: (S_0, X) contains “owner”
- delete a/a* from (S, X)
  - precondition: (S_0, X) contains “owner” or (S_0, S) contains “control”
- create object X
  - S_0 becomes owner of X
- destroy object X
  - precondition: (S_0, X) contains “owner”
- create subject S
  - S0 becomes “owner” of and has “control” on S
- destroy subject S
  - precondition: (S_0, S) contains “owner”

Example: UNIX File Access Control


rwxrw----
Owner can read, write, and execute the file
Any user in the owner's group can read and write the file
Users outside the group cannot read, write, or execute the file.

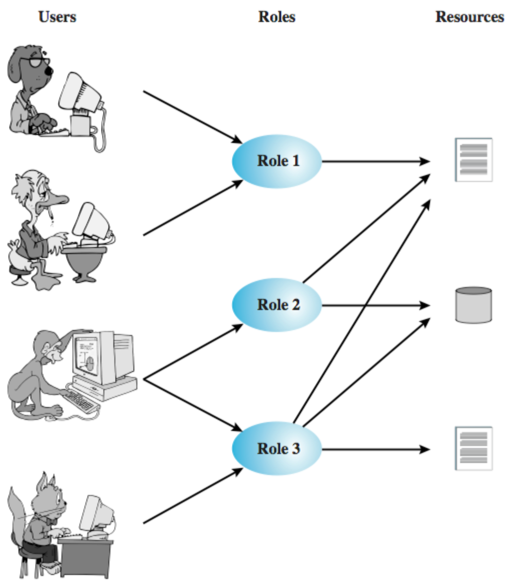
Example: UNIX File Access Control

- "set user ID"(SetUID) or "set group ID"(SetGID)
  - system temporarily uses rights of the file owner / group in addition to the real user's rights when making access control decisions
  - enables privileged programs to access files / resources not generally accessible
- sticky bit
  - on directory limits rename/move/delete to owner
- superuser
  - is exempt from usual access control restrictions

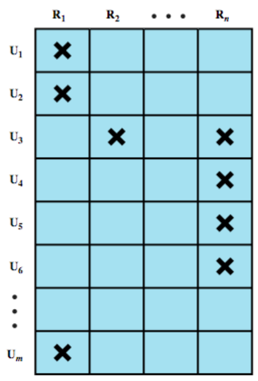
Example: UNIX Access Control Lists

- modern UNIX systems support ACLs
- can specify any number of additional users / groups and associated rwx permissions
- ACLs are optional extensions to std perms
- group perms also set max ACL perms
- when access is required
  - select most appropriate ACL
    - owner, named users, owning / named groups, others
  - check if have sufficient permissions for access

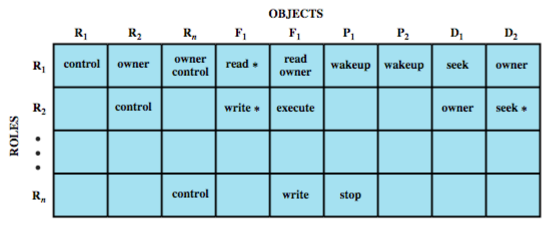
RBAC vs. DAC



RBAC vs. DAC



	R_1	R_2	• • •	R_n
U_1	✗			
U_2	✗			
U_3		✗		✗
U_4				✗
U_5				✗
U_6				✗
⋮				
U_m	✗			



OBJECTS

ROLES	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
⋮									
R_n			control		write	stop			

Extensions: Role Hierarchies

- provides the means to reflect the hierarchical structure of roles in an organization
- make use of the concept of inheritance
  - a superior (child) role inherits all the rights of the subordinate (parent) role, and it may have more rights
  - may support multiple inheritance or limited to strict inheritance tree

Extensions: Constraints

- constraints define relationships among roles or conditions related to roles
- examples:
  - mutual exclusion:

    a) set of roles such that a user can take only a single role in the set

    b) set of roles such that any access right can be granted to only one role in the set
  - cardinality
    - max number of users who can take a role
    - max number of roles that can be taken (or activated for a given session) by a user
    - max number of roles that can have a given access right
  - prerequisite roles
    - a user can take a given role only if she already has some other prerequisite roles (strict hierarchies, least privilege principle)

Computer Security: Principles and Practice

Chapter 10 – Trusted Computing and Multilevel Security

First Edition
by William Stallings and Lawrie Brown

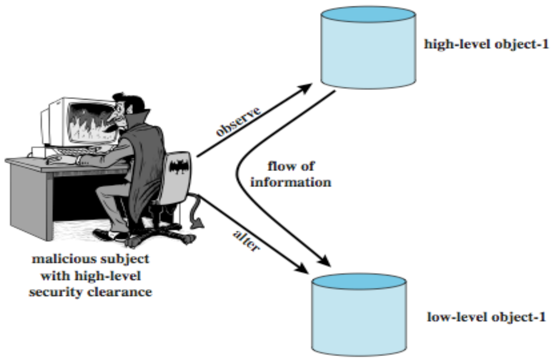
Lecture slides by Lawrie Brown

Bell-LaPadula (BLP) Model

- developed in 1970s as a formal access control model (MAC + DAC)
- subjects and objects have a **security class**
  - top secret > secret > confidential > unclassified
  - subject has a **security clearance** level
  - object has a **security classification** level
  - these levels control how a subject may access an object

Requirements for Multi-Level Security

- prevent information flow from higher levels to lower levels
  →no read up AND no write down



Motivation for Formal Models

- two fundamental computer security facts:
  - all complex software systems have flaw/bugs
  - it is extremely difficult to build computer hardware/software not vulnerable to attacks
- hence desire to prove that the design and the implementation satisfy security requirements
- led to development of formal security models
  - initially funded by US DoD
- Bell-LaPadula (BLP) became very influential

BLP Formal Description

- current state of system (*b*, *M*, *f*, *H*):
  - b – current access set, triplets of form (subject, object, access mode)
  - M – Access Matrix, represents current set of permissions
  - f – level function, assigns security level to each object and subject
  - H – hierarchy, a directed rooted tree whose nodes are objects in the system, and the security level of an object dominates the security level of its parent

BLP operations

1. Get access: Add a triple (*subject*, *object*, *access-mode*) to the current access set *b*, used by a subject to initiate access to an object in the requested mode.

2. Release access: Remove a triple (*subject*, *object*, *access-mode*) from the current access set *b*.

3. Change object level: Change the value of $f_o(O_j)$ for some object O_j. Used by a subject to alter the security level of an object.

4. Change current level: Change the value of $f_c(S_i)$ for some subject S_i. Used by a subject to alter the security level of a subject.

5. Give access permission: Add an access mode to some entry of the access permission matrix *M*. Used by a subject to grant an access mode on an object to another subject.

6. Rescind access permission: Delete an access mode from some entry of *M*. Used by a subject to revoke an access previously granted.

7. Create an object: Attach an object to the current tree structure *H* as a leaf. Used to create a new object.

8. Delete a group of objects: Detach from *H* an object and all other objects beneath it in the hierarchy. This operation may also modify the current access set *b* because all accesses to the object are released.

Note: Rules 1 and 2 alter the current access; rules 3 and 4 alter the level functions; rules 5 and 6 alter access permission; and rules 7 and 8 alter the hierarchy.

BLP properties

- three BLP properties:
  - simple security (ss) property
  - *-property
  - discretionary security (ds) property

ss-property: for all (S_i, O_j, read) in b, $f_c(S_i) \geq f_o(O_j)$.

*-property: for all (S_i, O_j, append) in B, $f_c(S_i) \leq f_o(O_j)$ and for all (S_i, O_j, write) in b, $f_c(S_i) = f_o(O_j)$

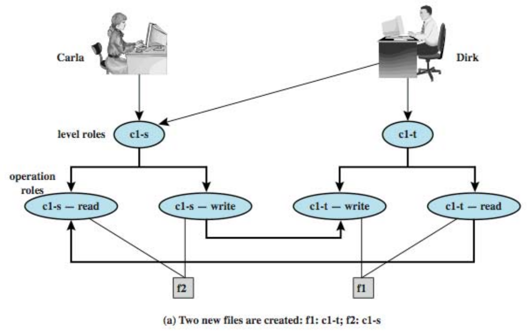
ds-property: for all (S_i, O_j, A_x) in b, $A_x \in M[S_i, O_j]$

Definition and proof of security

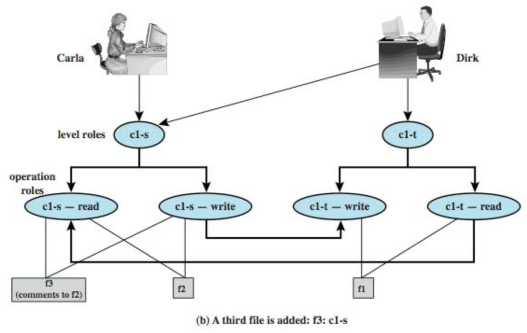
- The three BLP properties can be used to define a secure system as:
  1. initial state of the system satisfies the BLP properties
  2. every state change preserves the BLP properties

- in theory, it is possible to prove the system is secure (above properties are satisfied)
- in practice, for large systems, such proofs are usually not feasible

BLP Example

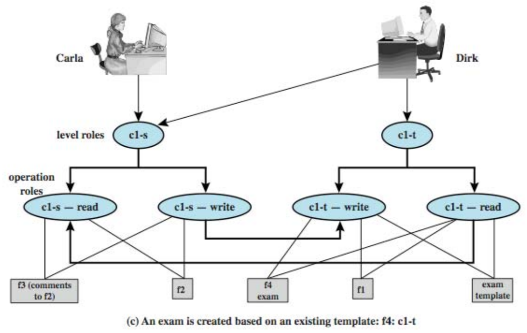


(a) Two new files are created: f1: c1-t; f2: c1-s

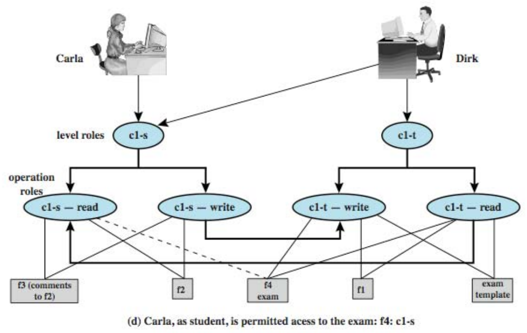


(b) A third file is added: f3: c1-s

BLP Example cont.

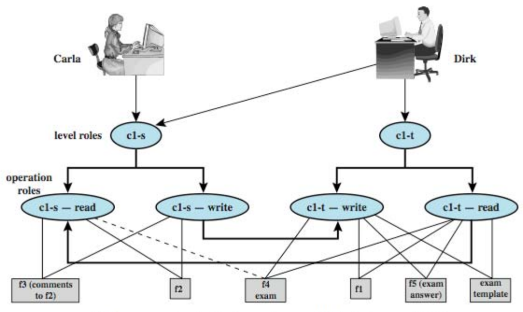


(c) An exam is created based on an existing template: f4: c1-t



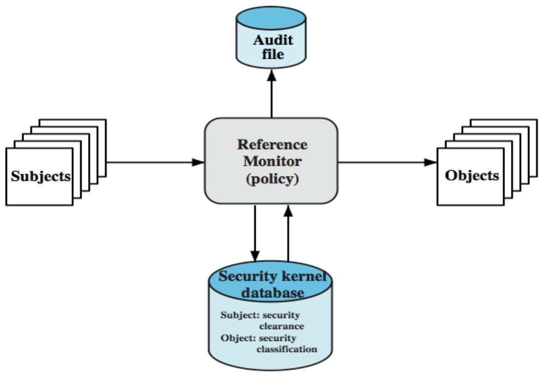
(d) Carla, as student, is permitted acess to the exam: f4: c1-s

BLP Example cont.



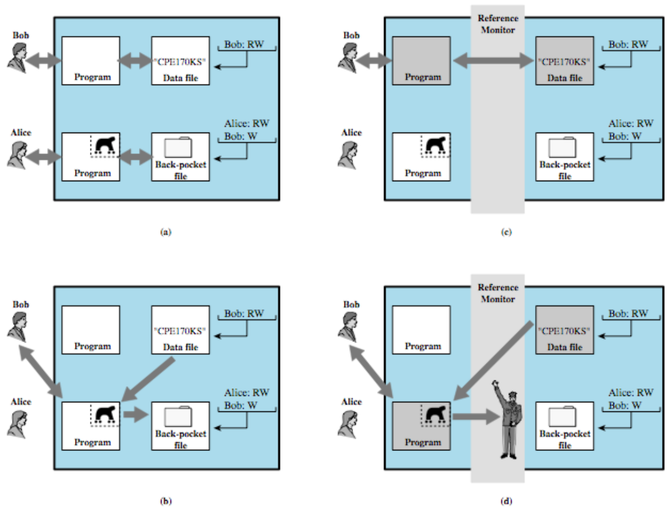
(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

Reference Monitors


Audit file
Subjects
Reference Monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

- the reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object

Example: Trojan Horse Defence



Bob
Alice
Program
"CPE170KS" Data file
Back-pocket file
Bob: RW
Alice: RW
Bob: W
Reference Monitor
(a)
(b)
(c)
(d)