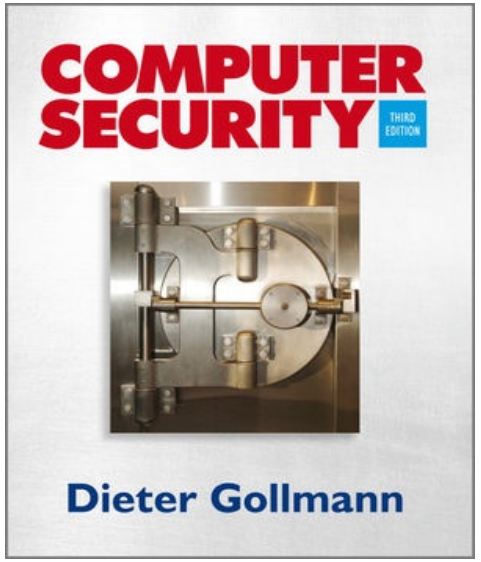
502049 – Introduction to Information Security

Chapter 4: Access Control



Ngoc-Tu Huynh, PhD
huynhngoctu@tdtu.edu.vn

Introduction

- Access control: who is allowed to do what?
- Traditionally, “who” is a person.
- Traditionally, “what” consists of an operation (read, write, execute, …) performed on a resource (file, directory, network port, …)
- The type of access control found in Unix, Windows.
- Today, access control is a more general task.
- Java sandbox: “who” is code running on a machine.

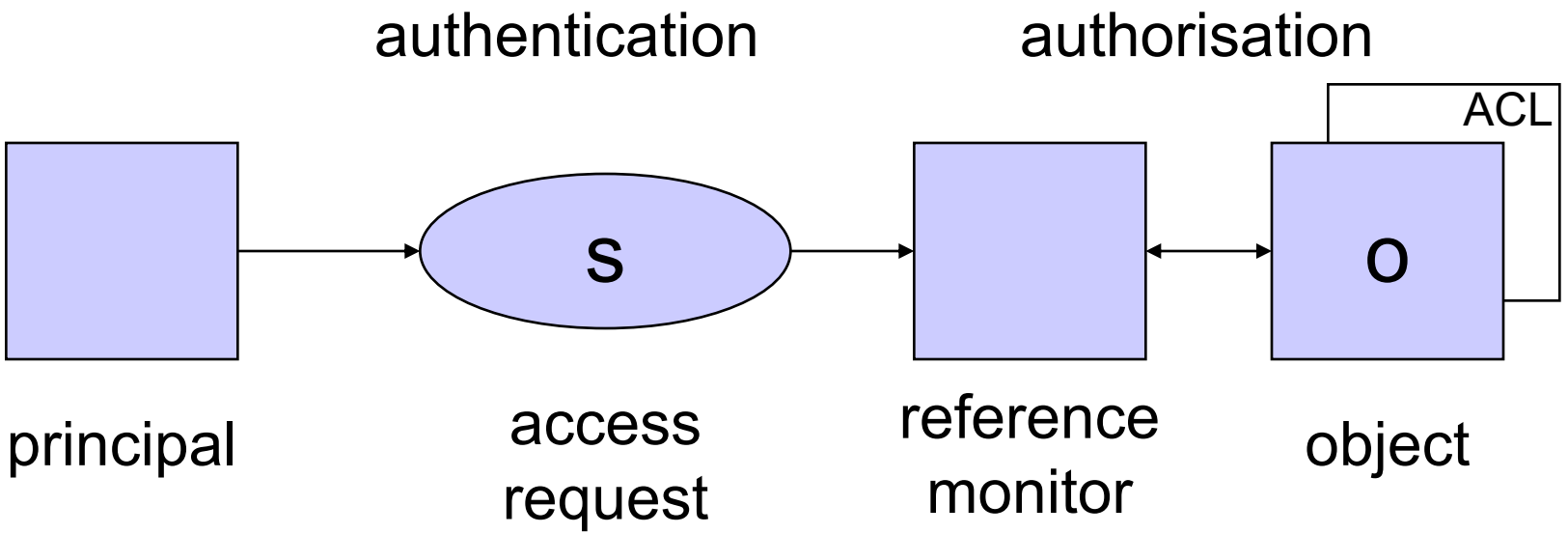
Agenda

- Fundamental terminology
  - Principals & subjects, access operations
- Authentication & authorisation
- Policies
  - Capabilities & access control list
  - Discretionary & mandatory access control
  - Role Based Access Control
  - Policy instantiation
- Structuring policies
  - Partial orderings & lattices

Security Policies

- Access control enforces operational security policies.
- A policy specifies who is allowed to do what.
- The active entity requesting access to a resource is called principal.
- The resource access is requested for is called object.
- Reference monitor is the abstract machine enforcing access control; guard mediating all access requests.
- Traditionally, policies refer to the requestor's identity and decisions are binary (yes/no).

Authentication & Authorisation



B. Lampson, M. Abadi, M. Burrows, E. Wobber: Authentication in Distributed Systems: Theory and Practice, ACM Transactions on Computer Systems, 10(4), pages 265-310, 1992

Authentication & Authorisation

- Authentication: reference monitor verifies the identity of the principal making the request.
  - A user identity is one example for a principal.
- Authorisation: reference monitor decides whether access is granted or denied.
- Reference monitor has to find and evaluate the security policy relevant for the given request.
- “Easy” in centralized systems.
- In distributed systems,
  - how to find all relevant policies?
  - how to make decisions if policies may be missing?

Authentication

- User enters username and password.
- If the values entered are correct, the user is “authenticated”.
- We could say: “The machine now runs on behalf of the user”.
- This might be intuitive, but it is imprecise.
- Log on creates a process that runs with access rights assigned to the user.
- Typically, the process runs under the user identity of the user who has logged on.

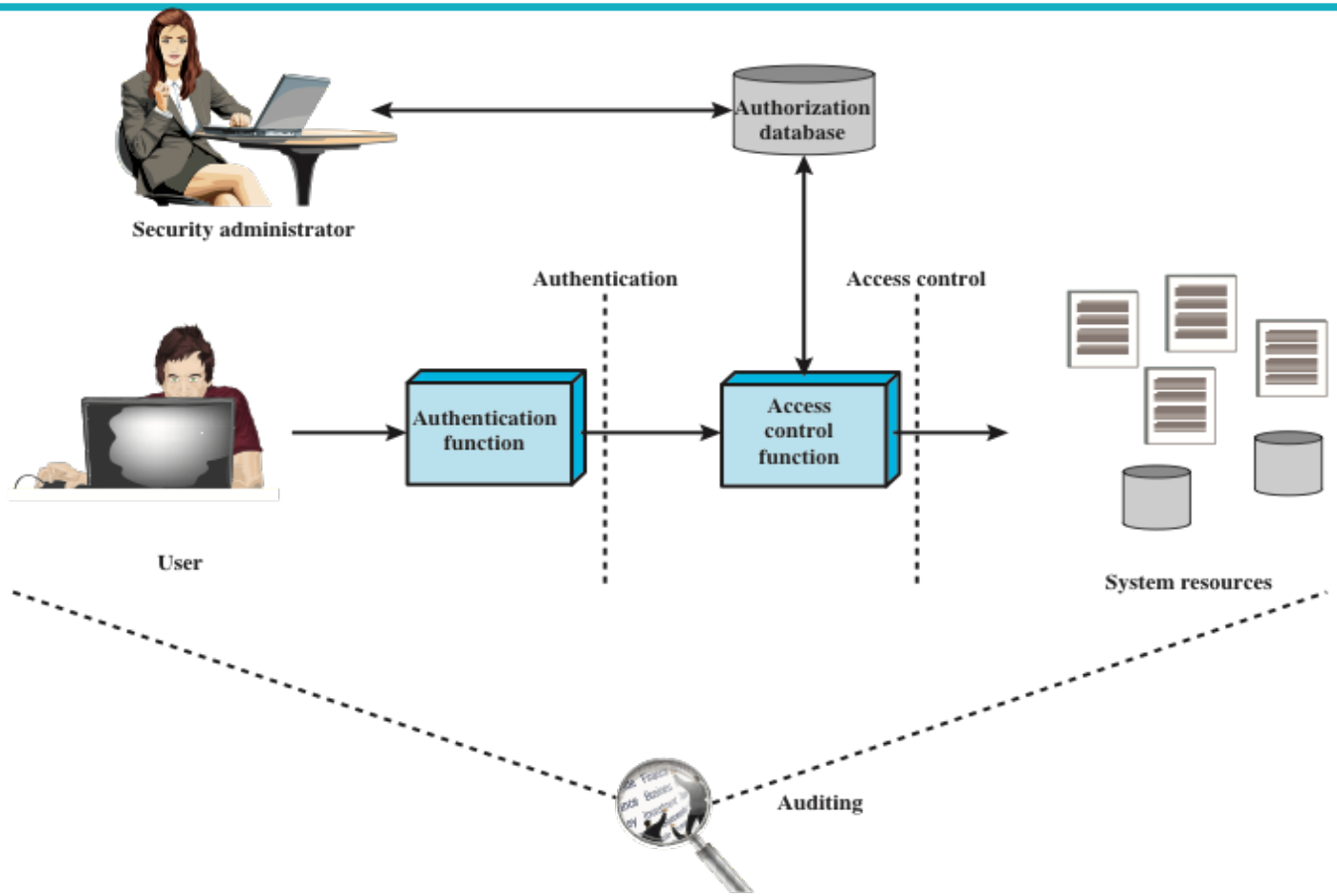


Figure 4.1 Relationship Among Access Control and Other Security Functions

Source: Based on [SAND94].

Users & User Identities

- Requests to reference monitor do not come directly from a user or a user identity, but from a process.
- In the language of access control, the process "speaks for" the user (identity).
- The active entity making a request within the system is called the subject.
- You must distinguish between three concepts:
  - User: person;
  - User identity (principal): name used in the system, possibly associated with a user;
  - Process (subject) running under a given user identity.

Principals & Subjects

- Terminology (widely but not universally adopted):
- Policy: A principal is an entity that can be granted access to objects or can make statements affecting access control decisions.
  - Example: user ID
- System: Subjects operate on behalf of (human users we call) principals; access is based on the principal's name bound to the subject in some unforgeable manner at authentication time.
  - Example: process (running under a user ID)

Principals & Subjects

- 'Principal' and 'subject' are both used to denote the entity making an access request.
- The term 'principal' is used in different meanings, which can cause much confusion.
  - M. Gasser (1990): Because access control structures identify principals, it is important that principal names be globally unique, human-readable and memorable, easily and reliably associated with known people.
  - We will examine later whether this advice is still valid.

Basic Terminology – Recap

- Subject/Principal: Active entity – user or process.
- Object: Passive entity – file or resource.
- Access operations: Vary from basic memory access (read, write) to method calls in object-oriented systems.
- Comparable systems may use different access operations or attach different meanings to operations which appear to be the same.

Access Operations

- Access right: right to perform an (access) operation; we will use the terms interchangeably.
- Permission: typically a synonym for access right.
- Privilege: typically a set of access rights given directly to roles like administrator, operator, ...

- These terms may have specific meanings in individual systems.

Access Operations

- On the most elementary level, a subject may
  - observe an object, or
  - alter an object.
- Some fundamental policies can be expressed with these basic access modes.
- For practical purposes a richer set of operations is more convenient.
- We will give examples for richer sets of access operations; note how certain terms are used with different meanings.

Subjects, Objects, and Access Rights

Subject

- An entity capable of accessing objects
- Three classes
  - Owner
  - Group
  - World

Object

- A resource to which access is controlled
- Entity used to contain and/or receive information

Access right

- Describes the way in which a subject may access an object
- Could include:
  - Read
  - Write
  - Execute
  - Delete
  - Create
  - Search

Elementary Access Operations

- Bell-LaPadula model (see chapter 11) has four access rights:
  - execute
  - read
  - append, also called blind write
  - write
- Mapping between access rights and access modes:

	execute	append	read	write
observe			X	X
alter		X		X

Rationale

- In a multi-user O/S, users open files to get access; files are opened for read or for write access so that the O/S can avoid conflicts like two users simultaneously writing to the same file.
- Write access usually includes read access; a user editing a file should not be asked to open it twice; Hence, write includes observe and alter mode.
- Few systems implement append; allowing users to alter an object without observing its content is rarely useful (exception: audit log).
- A file can be used without being opened (read); example: use of a cryptographic key; this can be expressed by an execute right that includes neither observe nor alter mode.

Access Rights (Unix/Linux)

- Three access operations on files:
  - read: from a file
  - write: to a file
  - execute: a file
- Access operations on directories:
  - read: list contents
  - write: create or rename files in the directory
  - execute: search directory
- Deleting files/subdirectories handled by access operations in the directory.

Administrative Access Rights

- Policies for creating and deleting files expressed by
  - access control on the directory (Unix).
  - specific create and delete rights (Windows, OpenVMS).
- Policies for defining security settings such as access rights handled by:
  - access control on the directory
  - specific rights like grant and revoke
- Rights in CORBA: get, set, use, manage

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Access Control Structures

Policy Focus

- Principals & objects provide a different focus of control:
  - What is the principal allowed to do?
  - What may be done with an object?
- Traditionally operating systems provide an infrastructure managing files and resources, i.e. objects; access control takes the second approach.
- Application oriented IT systems, like database management systems, provide services to the user and often control actions of principals.
- Note: some sources use authorisation to denote the process of setting policies.

Access Control Structures

- Policy is stored in an access control structure.
  - Access control structure should help to capture your desired access control policy.
  - You should be able to check that your policy has been captured correctly.
- Access rights can be defined individually for each combination of subject and object.
- For large numbers of subjects and objects, such structures are cumbersome to manage; intermediate levels of control are preferable.

Access Control Matrix

- At runtime, we could specify for each combination of subject and object the operations that are permitted.
  - S … set of subjects
  - O … set of objects
  - A … set of access operations
- Access control matrix: $M = (M_{so})_{s \in S, o \in O}$
- Matrix entry $M_{so} \subseteq A$ specifies the operations subject s may perform on object o.
- You can visualize the matrix as a (big) table.
- [B. Lampson: Protection, ACM OS Reviews,1974]

Access Control Matrix

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

(a) Access matrix

Figure 4.2 Example of Access Control Structures

Access Control Matrix

- Access control matrix has a row for each subject and a column for each object.

	bill.doc	edit.exe	fun.com
Alice	-	{exec}	{exec,read}
Bob	{read,write}	{exec}	{exec,read,write}

  - Access control matrix is an abstract concept,
  - not very suitable for direct implementation,
  - not very convenient for managing security.
- How do you answer the question: Has your security policy been implemented correctly?

Capabilities

- Focus on the subject
  - access rights stored with the subject
  - capabilities ≡ rows of the access control matrix

Alice	edit.exe: {exec}	fun.com: {exec,read}

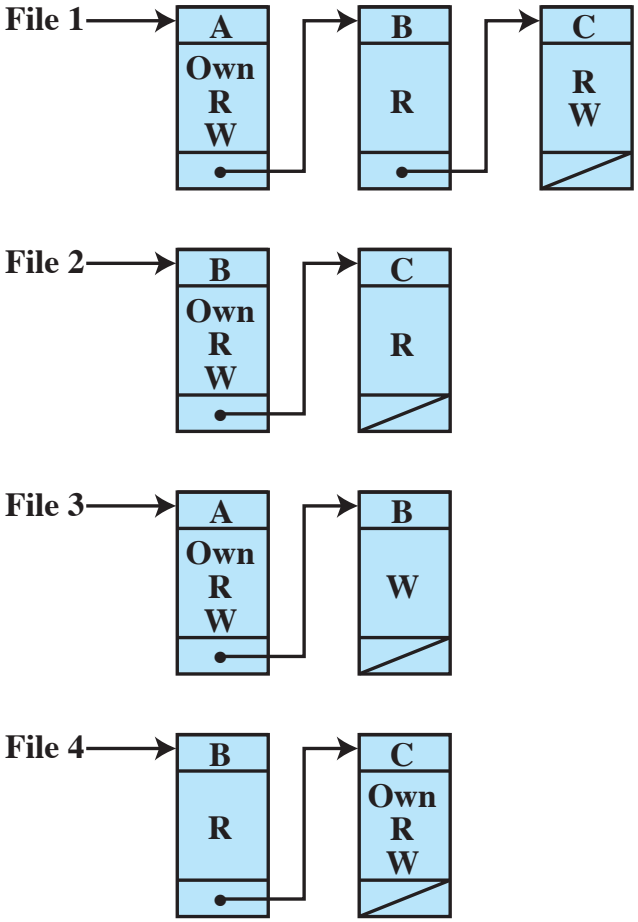
- Subjects may grant rights to other subjects; subjects may grant the right to grant rights.
- How to check who may access a specific object?
- How to revoke a capability?
- Distributed system security has created renewed interest in capabilities.

Access Control Lists (ACLs)

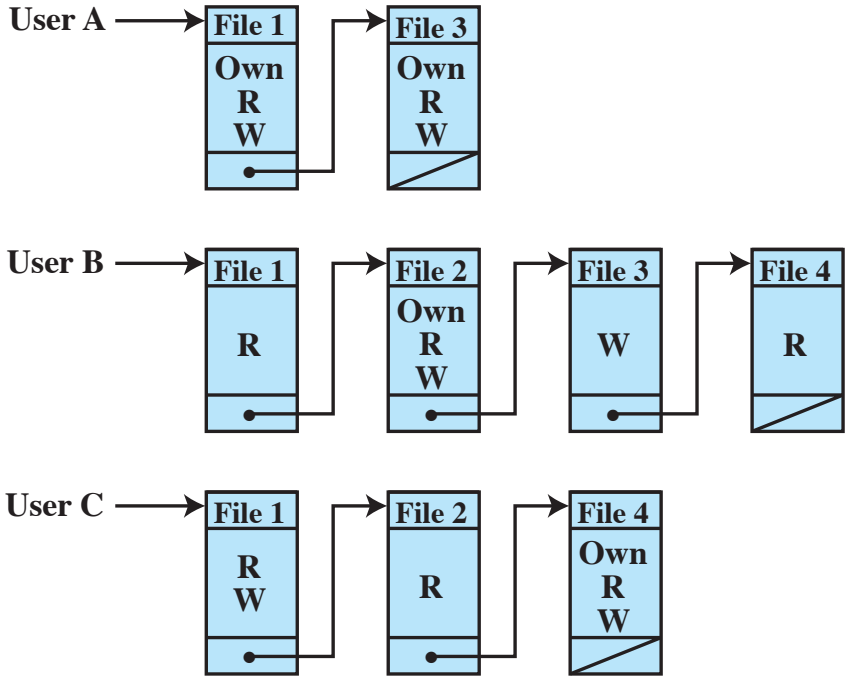
- Focus on the object
  - access rights of principals stored with the object
  - ACLs ≡ columns of the access control matrix

fun.com	Alice: {exec}	Bill: {exec,read,write}

- How to check access rights of a specific subject?
- ACLs implemented in most commercial operating systems but their actual use is limited.



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



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

Figure 4.2 Example of Access Control Structures

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

Table 4.2

Authorization Table for Files in Figure 4.2

(Table is on page 113 in the textbook)

		OBJECTS								
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Figure 4.3 Extended Access Control Matrix

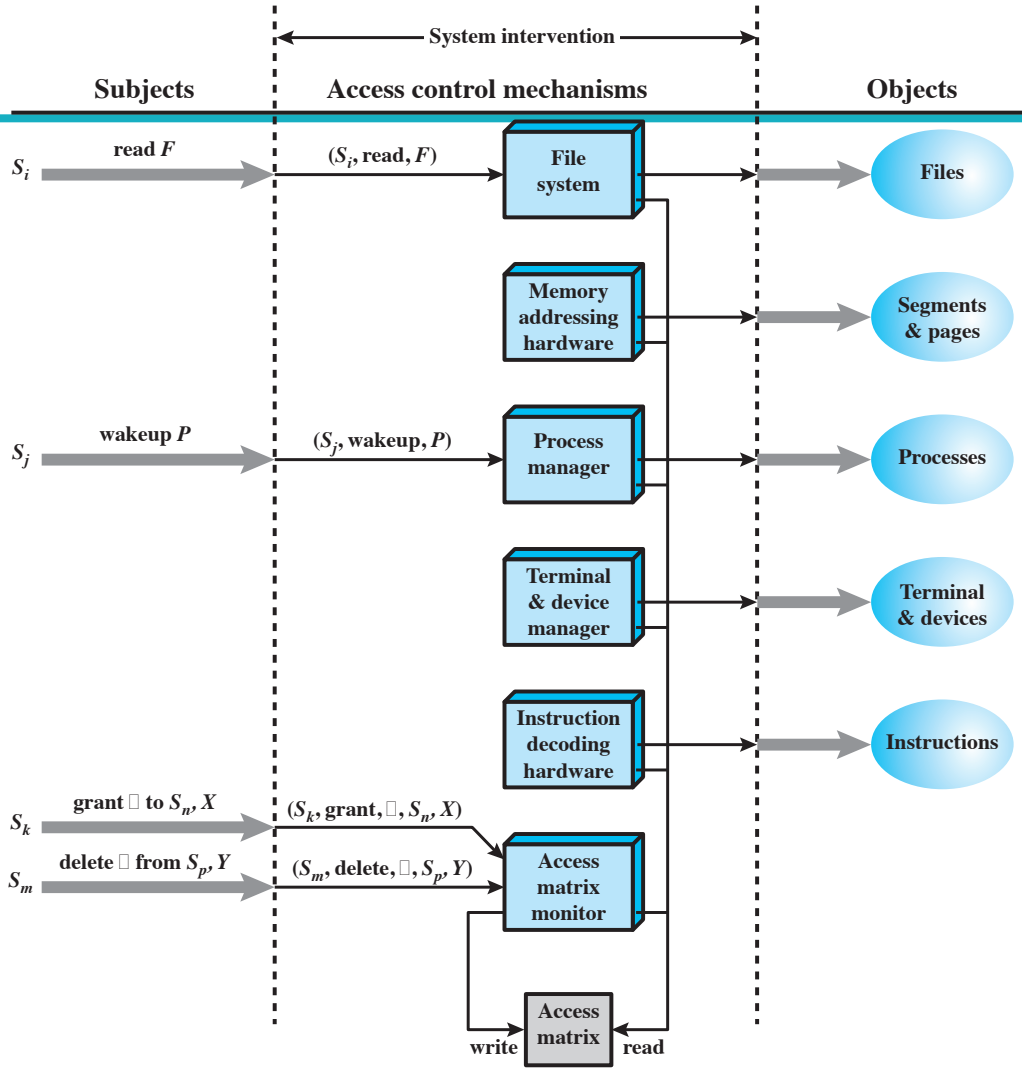


Figure 4.4 An Organization of the Access Control Function

Rule	Command (by S_o)	Authorization	Operation
R1	**transfer** $\left\{ \begin{matrix} \alpha* \\ \alpha \end{matrix} \right\}$ **to** S, X	'α*' in $A[S_o, X]$	store $\left\{ \begin{matrix} \alpha* \\ \alpha \end{matrix} \right\}$ in $A[S, X]$
R2	**grant** $\left\{ \begin{matrix} \alpha* \\ \alpha \end{matrix} \right\}$ **to** S, X	'owner' in $A[S_o, X]$	store $\left\{ \begin{matrix} \alpha* \\ \alpha \end{matrix} \right\}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	delete α from $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$	copy $A[S, X]$ into w
R5	**create object** X	None	add column for X to A; store 'owner' in $A[S_o, X]$
R6	**destroy object** X	'owner' in $A[S_o, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_o, S]$	delete row for S from A; execute **destroy object** S

Table 4.3

Access Control System Commands

(Table is on page 116 in the textbook)

Who Sets the Policy?

- Security policies specify how principals are given access to objects.
- Responsibility for setting policy could be assigned to
  - the owner of a resource, who may decree who is allowed access; such policies are called discretionary as access control is at the owner's discretion.
  - a system wide policy decreeing who is allowed access; such policies are called mandatory.
- Warning: other interpretations of discretionary and mandatory access control exist.

DAC & MAC

- Access control based on policies that refer to user identities was historically (since the 1970s) called discretionary access control (DAC).
- Referring to individual users in a policy works best within closed organisations.
- Access control based on policies that refer to security labels (confidential, top secret, …) was historically called mandatory access control (MAC).
- DAC and MAC have survived in computer security text books, but not very much in the wild.

Intermediate Levels

- “In computer science, problems of complexity are solved by adding another level of indirection.”
  [David Wheeler]

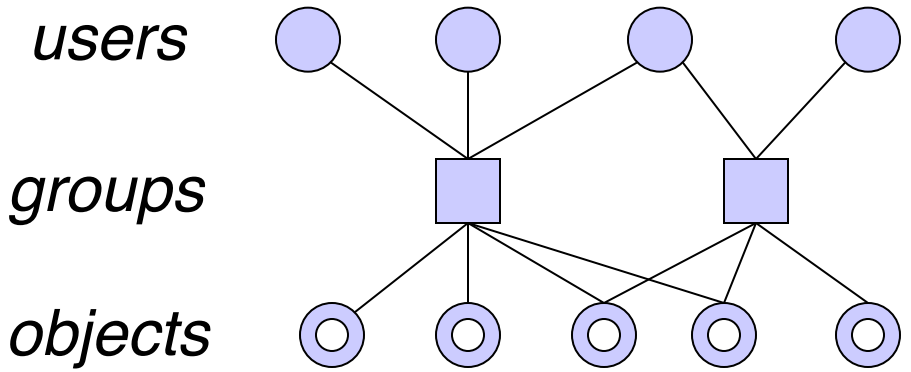
- We apply this principle and introduce intermediate layers between users and objects to represent policies in a more manageable fashion.

IBAC & Groups

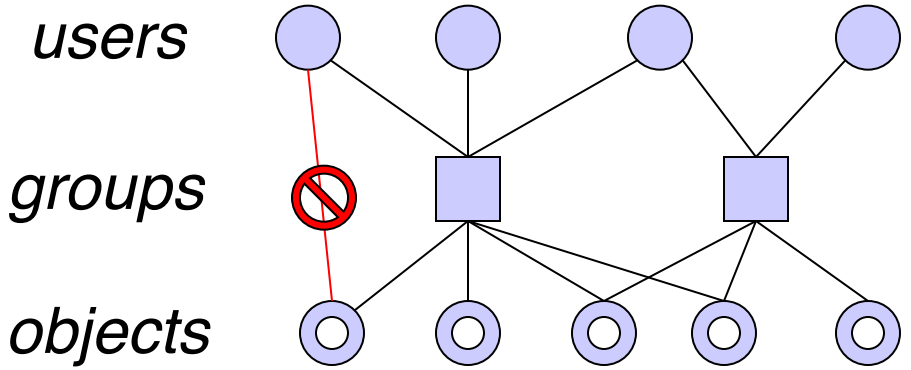
- We might use identity based access control (IBAC) instead of DAC.
- IBAC does not scale well and will incur an "identity management" overhead.
- Alice and Bob are students in a large class; teacher wants to give students access to some documents.
- Putting all names into several ACLs is tedious so the teacher defines a group, declares the students to be members of group, and puts group into the ACLs.
- Access rights are often defined for groups:
  - Unix: owner, group, others

Groups & Negative Permissions

Groups: intermediate layer between users and objects.



To handle exceptions, negative permissions withdraw rights



Roles

- Alternatively, in our example we could have created a role ‘student’.
- Definition: A role is a collection of procedures assigned to users; a user can have more than one role and more than one user can have the same role.
- Teacher creates a procedure for reading course material, assigns this procedure to the role ‘student’.
- A role ‘course tutor’ could be assigned a procedure for updating documents.

RBAC

- Role Based Access Control
- Procedures: 'High level' access operations with a more complex semantic than read or write; procedures can only be applied to objects of certain data types.
- Example: Funds transfer between bank accounts.
- Roles are a good match for typical access control requirements in business.
- RBAC typical found at the application level.
- Difference between groups and roles??

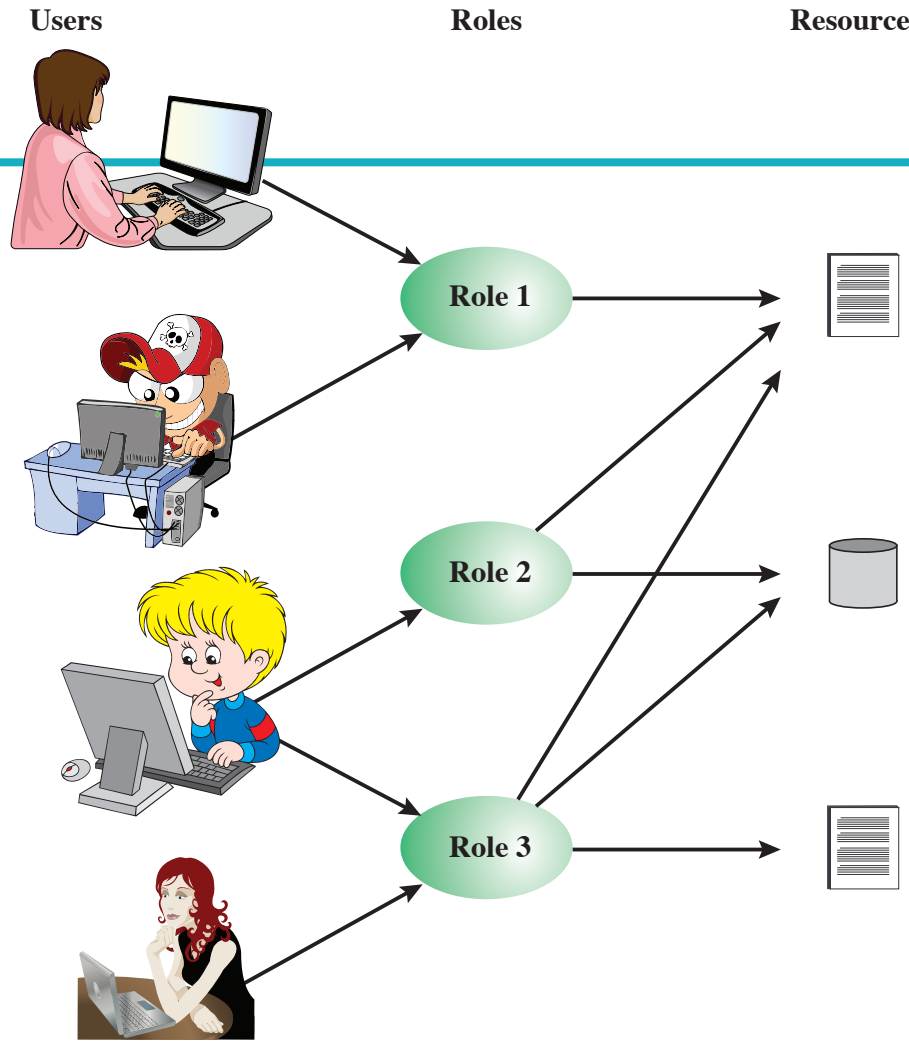
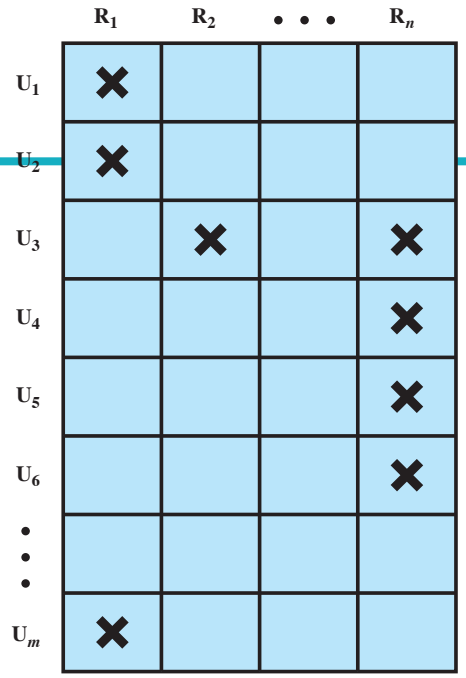


Figure 4.6 Users, Roles, and Resources



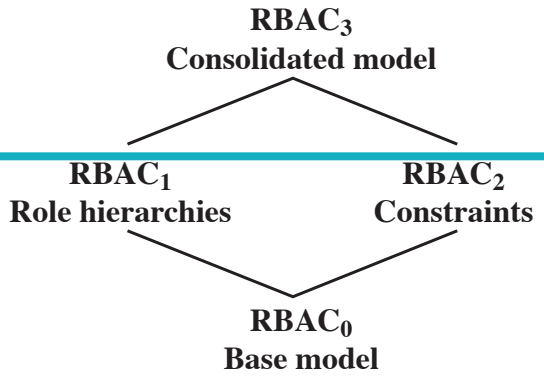
OBJECTS

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			

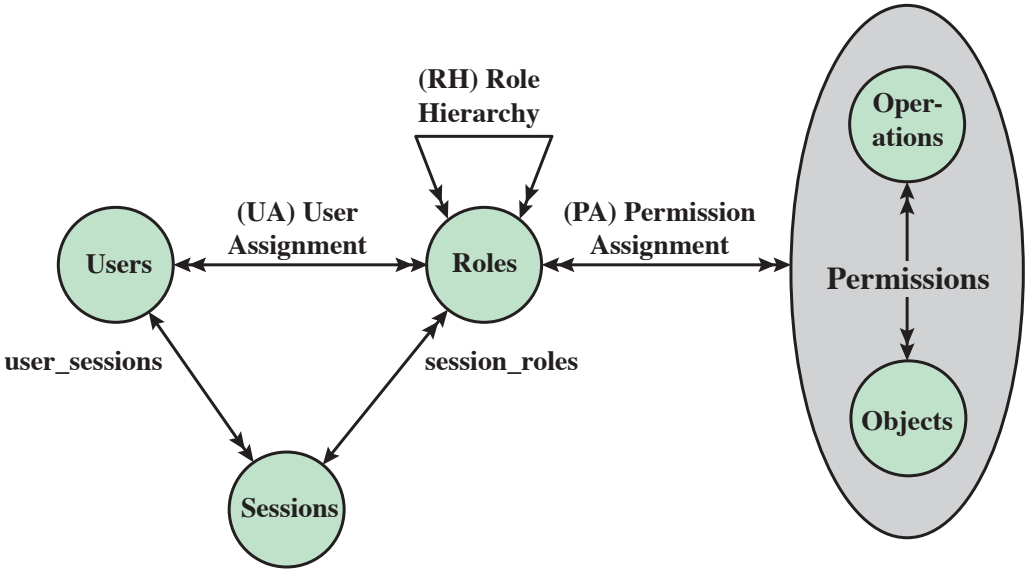
Figure 4.7 Access Control Matrix Representation of RBAC

More on RBAC

- Role hierarchies define relationships between roles; senior role has all access rights of the junior role.
- Do not confuse the role hierarchy with the hierarchy of positions (superior – subordinate) in an organisation.
- These two hierarchies need not correspond.
- Separation of duties is an important security principle; numerous flavours of static and dynamic separation of duties policies exist.
- Example: a manager is given the right to assign access rights to subordinates, but not the right to exercise those access rights.



(a) Relationship among RBAC models



(b) RBAC models

Figure 4.8 A Family of Role-Based Access Control Models.

Table 4.4
Scope RBAC Models

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

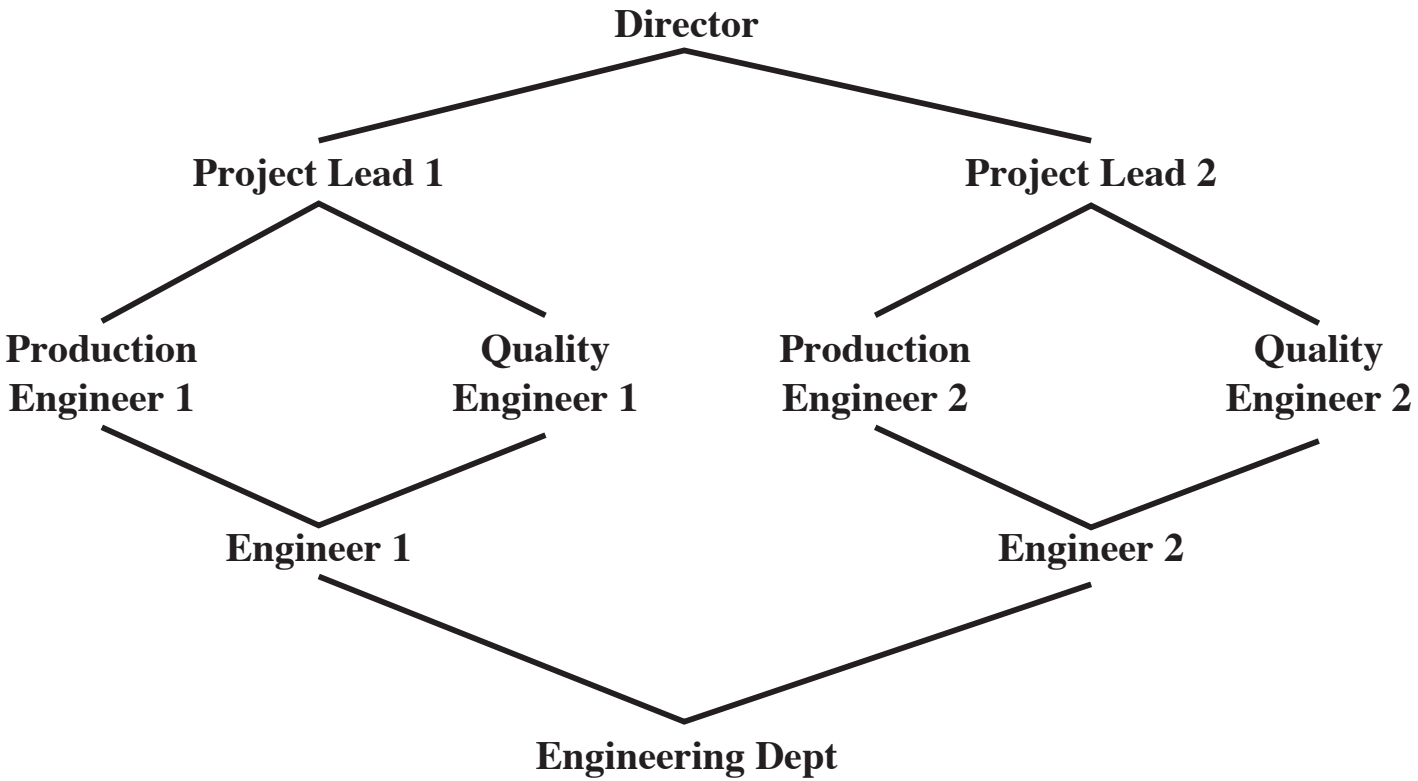


Figure 4.9 Example of Role Hierarchy

NIST: RBAC Levels

- Flat RBAC:
  - users are assigned to roles,
  - permissions are assigned to roles,
  - users get permissions via role membership;
  - support for user-role reviews.
- Hierarchical RBAC: adds support for role hierarchies.
- Constrained RBAC: adds separation of duties.
- Symmetric RBAC: support for permission-role reviews (can be difficult to provide in large distributed systems).

Role Based Access Control

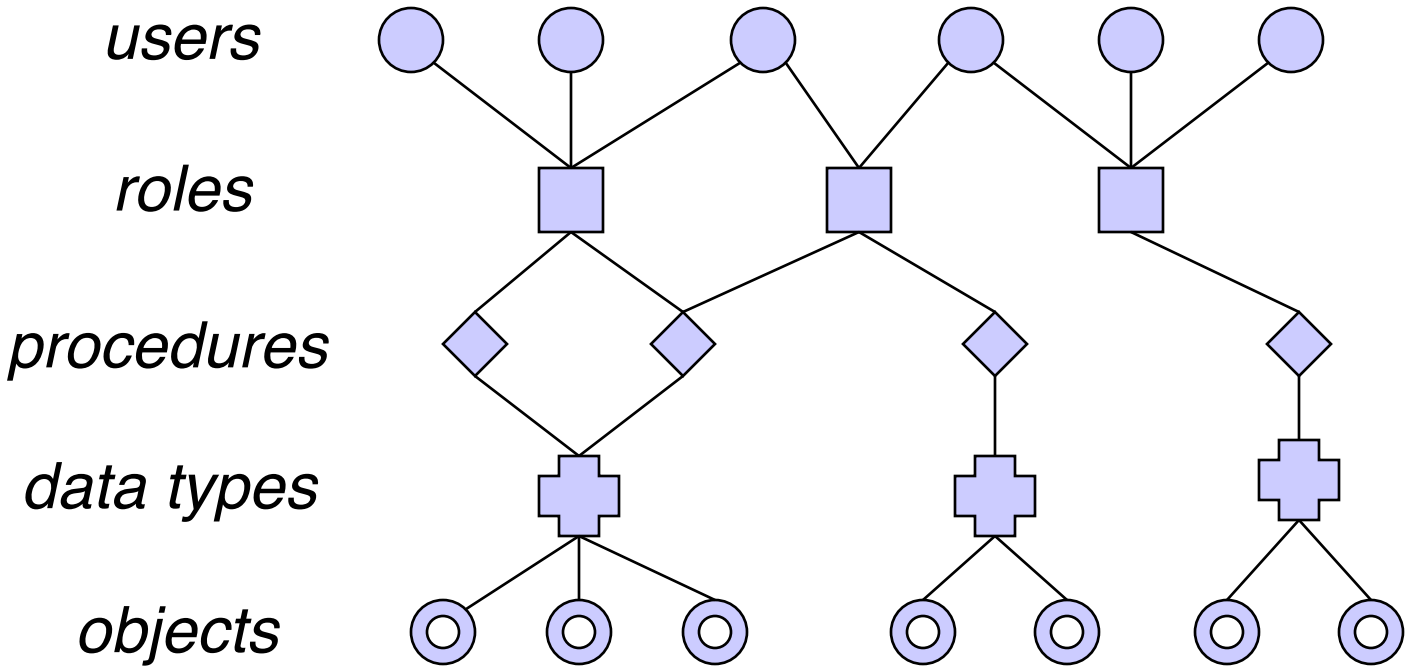
- Standard: American National Standards Institute: Role Based Access Control, ANSI-INCITS 359-2004.
- However,

  The term RBAC itself does not have a generally accepted meaning, and it is used in different ways by different vendors and users.

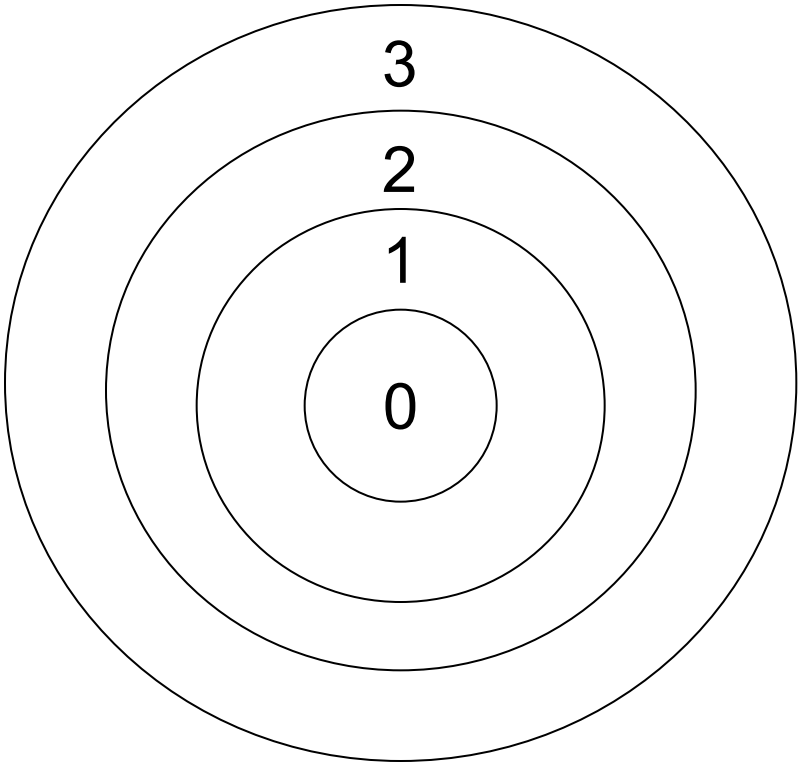
  [R. Sandhu, D. Ferraiolo, and R. Kuhn: *The NIST Model for Role-Based Access Control: Towards a Unified Standard*, Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, July 26-27, 2000

Lesson: Intermediate Controls

Intermediate controls for better security management; to deal with complexity, introduce more levels of indirection.



Protection Rings



Protection rings are mainly used for integrity protection.

Protection Rings

- Each subject (process) and each object is assigned a number, depending on its 'importance', e.g.
  - 0 – operating system kernel
  - 1 – operating system
  - 2 – utilities
  - 3 – user processes
- Numbers correspond to concentric protection rings, ring 0 in centre gives highest degree of protection.
- If a process is assigned number *i*, we say the process "runs in ring *i*".
- Access control decisions are made by comparing the subject's and object's ring.

Policy Instantiation

- When developing software you will hardly know who will eventually make use of it.
- At this stage, security policies cannot refer to specific user identities.
- A customer deploying the software may know its "authorized" users and can instantiate a generic policy with their respective user identities.
- Generic policies will refer to 'placeholder' principals like owner, group, others (world, everyone).
- Reference monitor resolves values of 'placeholders' to user identities when processing an actual request.

Structuring Policies

Structuring Access Control

- Some resources in an academic department can be accessed by all students, other resources only by students in a particular year.
- Department creates groups like 'All-Students' and 'Y1-Students'.
- The two groups are related, Y1-Students is a subgroup of All-Students; if All-Students has access to a resource, so has Y1-Students.
- No such direct relationship between Y1-Students and Y2-Students.

Partial Orderings

- We now can use comparisons in security policies: Is the user's group a subgroup of the group permitted to access this resource?
- Some groups are related but others are not (e.g. Y1-Students and Y2-Students).
- Relationships are transitive: CS101-Students $\subseteq$ Y1-Students $\subseteq$ All-Students
- In mathematical terms, we are dealing with a partial ordering.

Mathematical Definition

- A partial ordering $\leq$ ('less or equal') on a set L is relation on $L \times L$ that is
  - reflexive: for all $a \in L$, $a \leq a$
  - transitive: for all $a,b,c \in L$, if $a \leq b$ and $b \leq c$, then $a \leq c$
  - antisymmetric: for all $a,b \in L$, if $a \leq b$ and $b \leq a$, then $a = b$
- If $a \leq b$, we say 'b dominates a' or 'a is dominated by b'.

Examples

- Integers with the relation “divides by”:
  - We can order 3 and 6 (3 divides 6); we cannot order 4 and 6.
- Integers with the relation $\leq$ (“less or equal”):
  - We can order any two elements (total ordering).
- Strings with the prefix relation:
  - We can order AA and AABC (AA is a prefix of AABC) but not AA and AB.
- Power set $P(C)$ with subset relation $\subseteq$:
  - We can order $\{a,b\}$ and $\{a,b,c\}$ ($\{a,b\} \subseteq \{a,b,c\}$) but not $\{a,b\}$ and $\{a,c\}$.

Example: VSTa Microkernel

- Groups in Unix are defined by their group ID and are not ordered.
- VSTa uses (cap)abilities to support hierarchies: VSTa (cap)ability is a list of integers $.i_1.i_2. \cdots .i_n$, e.g. .1, .1.2, .1.2.3, .4, .10.0.0.5
- Abilities are ordered by the prefix relation:
  - a_2 is a prefix of a_1 (written as $a_2 \leq a_1$) if there exists a_3 so that $a_1 = a_2 a_3$.
  - The empty string ε is the prefix of any ability.
- For example: $.1 \leq .1.2 \leq .1.2.4$ but not $.1 \leq .4$!

Abilities and our Example

- Assign abilities to groups:
  - All-students: .3
  - Y1-Students: .3.1
  - CS101-Students: .3.1.101
  - CS105-Students .3.1.105
- Label objects with appropriate abilities
- Policy: access is given if the object's label is a prefix of the subject's label; CS101-Students have access to objects labelled .3.1.101 or .3.1 or .3 but not to objects labelled .3.1.105

Null Values

- Consider the dual of the previous policy: access is granted if the subject's ability is a prefix of the ability of the object.
- A subject without an ability has access to every object.
- Frequent problem: when an access control parameter is missing the policy is not evaluated and access is granted.
- NULL DACL problem in Windows: Nobody has access to a file with an empty ACL but everyone has access to a file with no ACL.

Towards Lattices

- In our example, how should we label objects that may be accessed both by CS101-Students and CS105-Students?
- Answer: ??
- How should we label a subject that may access resources earmarked for CS101-Students and resources earmarked forCS105-Students?
- Answer: ??
- To answer both questions, we need more structure than just partial orderings.

Towards Lattices

The slide on lattices to remember

- Assume that a subject may observe an object only if the subject's label is higher than the object's label. We can ask two questions:
  - Given two objects with different labels, what is the minimal label a subject must have to be allowed to observe both objects?
  - Given two subjects with different labels, what is the maximal label an object can have so that it still can be observed by both subjects?
- A lattice is a mathematical structure where both questions have unique 'best' answers.

Lattice $(L,\leq)$

The slide on lattices you must not memorize

- A lattice $(L,\leq)$ is a set L with a partial ordering $\leq$ so that for every two elements $a,b \in L$ there exists
  - a least upper bound $u \in L$: $a \leq u, b \leq u$, and for all $v \in L$: $(a \leq v \wedge b \leq v) \Rightarrow u \leq v$.
  - a greatest lower bound $l \in L$: $l \leq a, l \leq b$, and for all $k \in L$: $(k \leq a \wedge k \leq b) \Rightarrow k \leq l$.
- Lattices come naturally whenever one deals with hierarchical security attributes.

System Low & System High

- A label that is dominated by all other labels is called System Low.
- A label that dominates all other labels is called System High.
- System Low and System High need not exist; if they exist, they are unique.
- When *L* is a finite set, the elements System Low and System High exist.

Lattices - Example 1

- The natural numbers with the ordering relation ‘divides by’ form a lattice:
  - The l.u.b. of a,b is their least common multiple.
  - The g.l.b. of a,b is their greatest common divisor.
  - There exists an element System Low: the number 1.
  - There is no element System High.

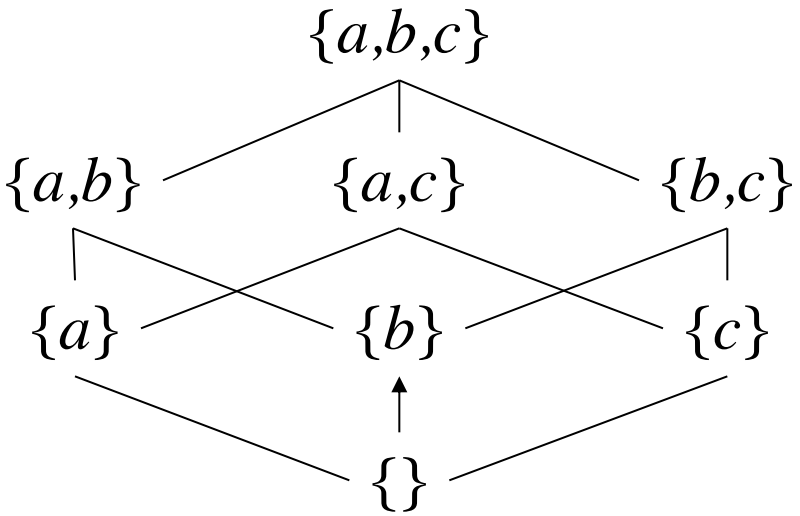
Lattices - Example 2

- The integers with the ordering $\leq$ form a lattice:
  - The l.u.b. of a,b is the maximum of a and b.
  - The g.l.b. of a,b is the minimum of a and b.
  - Elements System Low and System High do not exist.

- The integers with the ordering $\leq$ are a total ordering.

Lattices - Example 3

- $(P(\{a,b,c\}), \subseteq)$, i.e. the power set of $\{a,b,c\}$, with the subset relation as partial ordering:
  - least upper bound: union of two sets.
  - greatest lower bound: intersection of two sets.



Lines indicate the subset relation.

Summary

- Security terminology is ambiguous.
- Policies expressed in terms of principals and objects.
- In identity-based access control, users are principals.
- Deployed in practice: RBAC, ACLs to a minor extent.
- More sophisticated policies draw you into mathematics.
- We have covered ‘classical’ access control; we return to current trends later.
- Distinguish between access control as a security service and its various implementations.