Access Control

- Subject and objects
- Access operations
- Access control structures
- Intermediate controls
- Lattice of security levels (multi-level security)

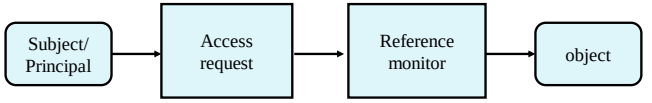
Motivation

Why do we need access control?

- Confidentiality, a user should be able to deny other users read access to his files
- Integrity, a user should be able to protect his files from modification or deletion by other users
- Help users to avoid unintentional change of important system files
- Help users to avoid unintentional change of important personal files, e.g., photos

Fundamental Model of Access Control

- *Subject* (active) users, processes, …
- *Object* (passive) files, resources, …
- *Access operation* read, write, …
- *Reference monitor* grants/denies access



Fundamental model of access control

- Subject/Principal
  - A *principal* is granted or denied access, e.g., UID
  - A *subject* acts on behalf of the principal, e.g., process running under a UID
  - *NOTE:* In some litterature only *subject* is used
- Subject/Object
  - A *subject* is the active party
  - An *object* is the passive party
  - Note that an entity can be subject in one request but object in another
- Access Right
  - Describes in which way a subject may access an object

Access Control can focus on one of two things:

1. What a subject is allowed to do
2. What may be done with an object

Access operations

Elementary level:

- Observe: look at the contents of an object
- Alter: change the contents of an object
- This is often too general to be practical

Bell-LaPadula security model:

- Execute, Append, Read, Write

	Execute	Append	Read	Write
Observe			X	X
Alter		X		X

Relation between Bell-LaPadula access rights and elementary access modes

In Unix it is different

- Write does not imply read

	object	
Access rights	**File**	**Directory**
read	Read file	List directory contents
write	Write file	Create/Delete/ Rename a file
execute	Execute program	Search directory

Security policy

- Discretionary access control – The owner of an object decides the access rights
- Mandatory access control – The system decides the access rights

- Orange Book:
  - Discretionary access control – Access is restricted based on the identity of the subject
  - Mandatory access control – Access is restricted based on the information sensitivity of an object and the authorization level of a subject.

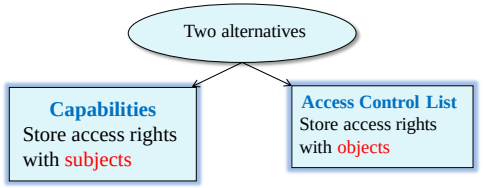
Access Control Matrix

- Access rights individually defined for each subject and object
- Let
  - S: the set of subjects
  - O: the set of objects
  - A: the set of access operations
- The access rights are uniquely defined by the *access control matrix*, $M=(M_{so})$ with $M_{so} \subseteq A$, $s \in S$, $o \in O$

	Bill.txt	Edit.exe	Prog.php
Alice	{read}	{execute}	{read,execute}
Bill	{read,write}	-	{read}
Charlie	{read}	-	-

Access Control Matrix

- Abstract concept
  - Size of matrix will be large
  - Much redundancy. (Many empty entries, many entries that are the same)
  - Creation and deletion of objects difficult to manage efficiently



Capabilities

- Separate each row

	Bill.txt	Edit.exe	Prog.php
Alice	{read}	{execute}	{read,execute}
Bill	{read,write}	-	{read}
Charlie	{read}	-	-

Alice's capability: Bill.txt: read; Edit.exe: execute; Prog.php:read,execute
Bill's capability: Bill.txt: read,write; Prog.php: read
Charlie's capability: Bill.txt: read

Difficult to determine who has access to a given object

Access Control List (ACL)

- Separate each column

	Bill.txt	Edit.exe	Prog.php
Alice	{read}	{execute}	{read,execute}
Bill	{read,write}	-	{read}
Charlie	{read}	-	-

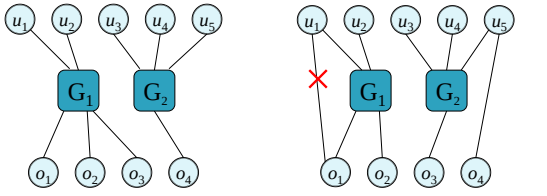
ACL for Bill.txt: Alice: read; Bill: read,write; Charlie: read;
ACL for Edit.exe: Alice: execute
ACL for Prog.php: Alice: read, execute; Bill: read

Difficult to get an overview of an individual user's permissions

Groups and Negative Permissions

- Putting users into a group can simplify

Beware of *policy conflicts!*



Should u_1 get access to o_1?

- Denying access may not be the same as not allowing access

Two important principles

- Principle of least privilege
  - A user or process should only have access to resources that are necessary
    - More stability – processes can not affect each other more than necessary and only affect a limited part of the system
    - More security – Vulnerabilities on one application can not be used to exploit other parts of the system
- Separation of duties
  - Security critical functionality must be performed by more than one user
  - Prevents fraud and errors
  - Sometimes difficult to achieve
  - **Example:** designer/implementer should not be same as tester
  - **Example 2**: Control of nuclear missile launch

Role-based access control (RBAC)

- Access rights are derived from a user's current role
- **Example:**

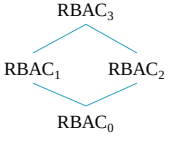
  User + current job → role
- Motivation: Users come and go, roles can be kept more static
- Common in database management systems

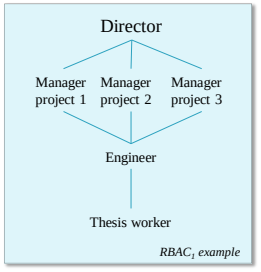
	Role 1	Role 2	Role 3	…	Role n
User 1	X				
User 2		X			
User 3	X		X		
User 4		X			
User 5		X			X
:					
User m		X	X		

	Object 1	Object 2	Object 3		Object k
Role 1	owner	modify	stop, start		copy
Role 2	append				
Role 3	read		start		
Role n	read				defrag

RBAC models

- **RBAC$_0$**: Base model
  - User, roles, permission, session
- **RBAC$_1$**: Role Hierarchies
  - Allow inheritance
- **RBAC$_2$**: Constraints
  - Mutually exclusive roles (separation of duties)
  - Cardinality (e.g., only one manager per project, only a certain number of roles for each user)
  - Prerequisite (You must have a subordinate role – allows implementation of least privilege)
- **RBAC$_3$**: RBAC$_1$ + RBAC$_2$
  - Combining hierarchies with constraints





Partial orderings

- Set of security levels L
- A partial ordering $\leq$ on a set L is a relation on L x L which is
  - *Reflexive:* $a \leq a$, for all $a \in L$
  - *Transitive:* if $a \leq b$ and $b \leq c$ then $a \leq c$, for all $a \in L$
  - *Antisymmetric:* if $a \leq b$ and $b \leq a$ then $a=b$, for all $a \in L$
- Example:
  - Powerset P(X) with subset relation as partial ordering

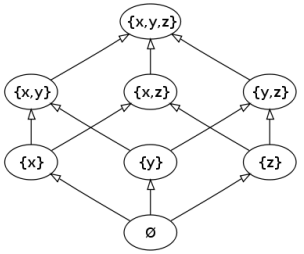
Powerset as partial ordering

- The powerset P(X) is the set of all subsets of the set X.
- Let X = {x,y,z}
- Then

  P(X) = { {∅}, {x}, {y}, {z}, {x,y}, {x,z}, {y,z}, {x,y,z} }
- We can define the partial ordering (P(X), ⊆)
- We have e.g.,
  - {x} ≤ {x,y}
  - {x,y} ≤ {x,y,z}
  - Note that there is no ordering between e.g., {x} and {y,z}
- We can say that a subject can access an object if object's label is a subset of the subject's label
  - Subject with label {x,y} can access object with label {x} since {x} ≤ {x,y}

Hasse diagram

- Graphical representation of a partially ordered set
- There is an edge between node a and b if and only if
  - a ≤ b and a ≠ b
  - There is no c ∈ L so that a ≤ c ≤ b and a ≠ c, b ≠ c
- Example: Hasse diagram of partially ordered set (P({x,y,z}), ⊆)

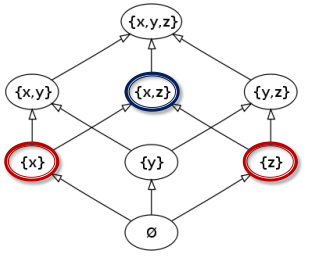


Lattice of security levels

- A *lattice* can answer two questions:
  - Given two objects at different security levels, what is the minimal security level a subject must have to access both?
  - Given two subjects at different security levels, what is the maximum security level an object can have so that it can be accessed by both subjects?
- Definition: A lattice $(L, \leq)$ consists of a set L and a partial ordering $\leq$. For $a, b \in L$ there is a least upper bound $u \in L$ and a greatest lower bound $l \in L$.
  - $a \leq u$, $b \leq u$, and $\forall v \in L : (a \leq v \wedge b \leq v) \Rightarrow (u \leq v)$
  - $l \leq a$, $l \leq b$, and $\forall k \in L : (k \leq a \wedge k \leq b) \Rightarrow (k \leq l)$
- We say that ***b*** dominates ***a*** if $a \leq b$
- Powerset with subset relation as partial ordering is a lattice

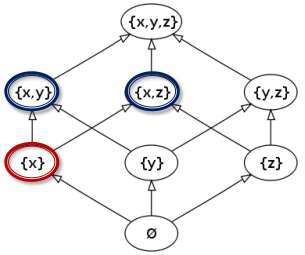
Subset relation is a lattice

- Example of least upper bound u
  - Let a = {x} and b = {z}
  - Then u = {x,z}
  - {x} ≤ {x,z} and {z} ≤ {x,z} and for all elements v such that $a \leq v$ and $b \leq v$ we also have $u \leq v$
  - In this case {x,z} and {x,y,z} are the only elements that dominates {x} and {z} and clearly {x,z} ≤ {x,z} and {x,z} ≤ {x,y,z}



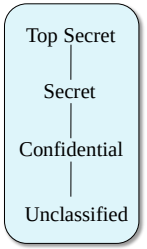
Subset relation is a lattice

- Example of greatest lower bound *l*
  - Let a = {x,y} and b = {x,z}
  - Then l = {x}
  - {x} ≤ {x,y} and {x} ≤ {x,z} and for all elements k such that $k \leq a$ and $k \leq b$ we also have $k \leq l$
  - In this case {x} and {∅} are the only elements that are dominated by {x,y} and {x,z} and clearly {x} ≤ {x} and {∅} ≤ {x}



Multilevel security

Linear ordering



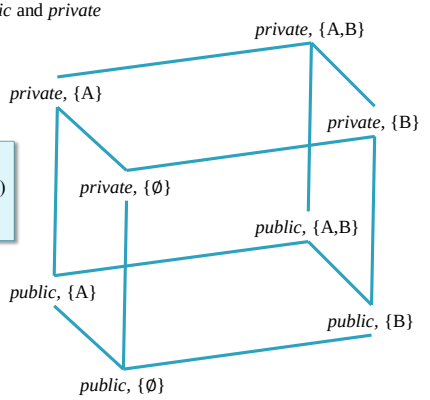
Combine with a set of categories to make it more flexible:

H is a set of classifications
C is a set of categories
Security level is (h,c), $h \in H$, $c \in C$
Partial ordering:
$(h_1,c_1) \leq (h_2,c_2)$ iff $h_1 \leq h_2$ and $c_1 \subseteq c_2$

Lattice with security labels

- Example of lattice
  - 2 classifications: *public* and *private*
  - 2 categories: *A* and *B*

(public, {A}) ≤ (private, {A})
(public, {B}) ≤ (public, {A,B})
(public, {B}) ≰ (private, {A})

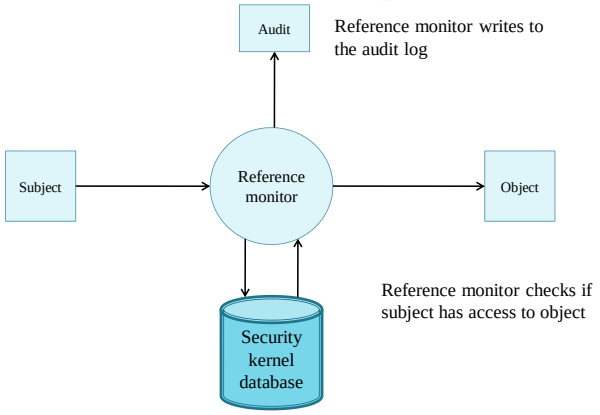


Reference Monitors

Three similar concepts

- **Reference monitor**: Abstract machinery that controls all access to objects.
- **Security kernel**: the hardware, software etc. that *implements* the reference monitor concept.
- **Trusted computing base (TCB):** The set of all protection mechanisms enforcing a security policy

Reference monitor concept



Reference monitor

- Requirements
  - Tamper proof
  - Must always be invoked
  - Small to allow analysis
- Where should we place it?
  - Hardware
  - Operating system
  - Service layer (e.g., JVM)
  - Application

Controlled invocation

- A user wants to execute an operation requiring specific access rights (which the user does not have)
  - supervisor mode.
- Use an API in order to execute the operation
- The system only performs a predefined set of operations