Access Control

(Another) definition of computer security

Measures to implement and assure security services in a computer system, particularly those that assure **access control** service. RFC 4949 Internet Security Glossary

(A) definition of access control

A process by which use of system resources is regulated according to a **security policy** and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy. RFC 4949 Internet Security Glossary

The security policy specifies who or what may have access to each system resource, and the type of access.

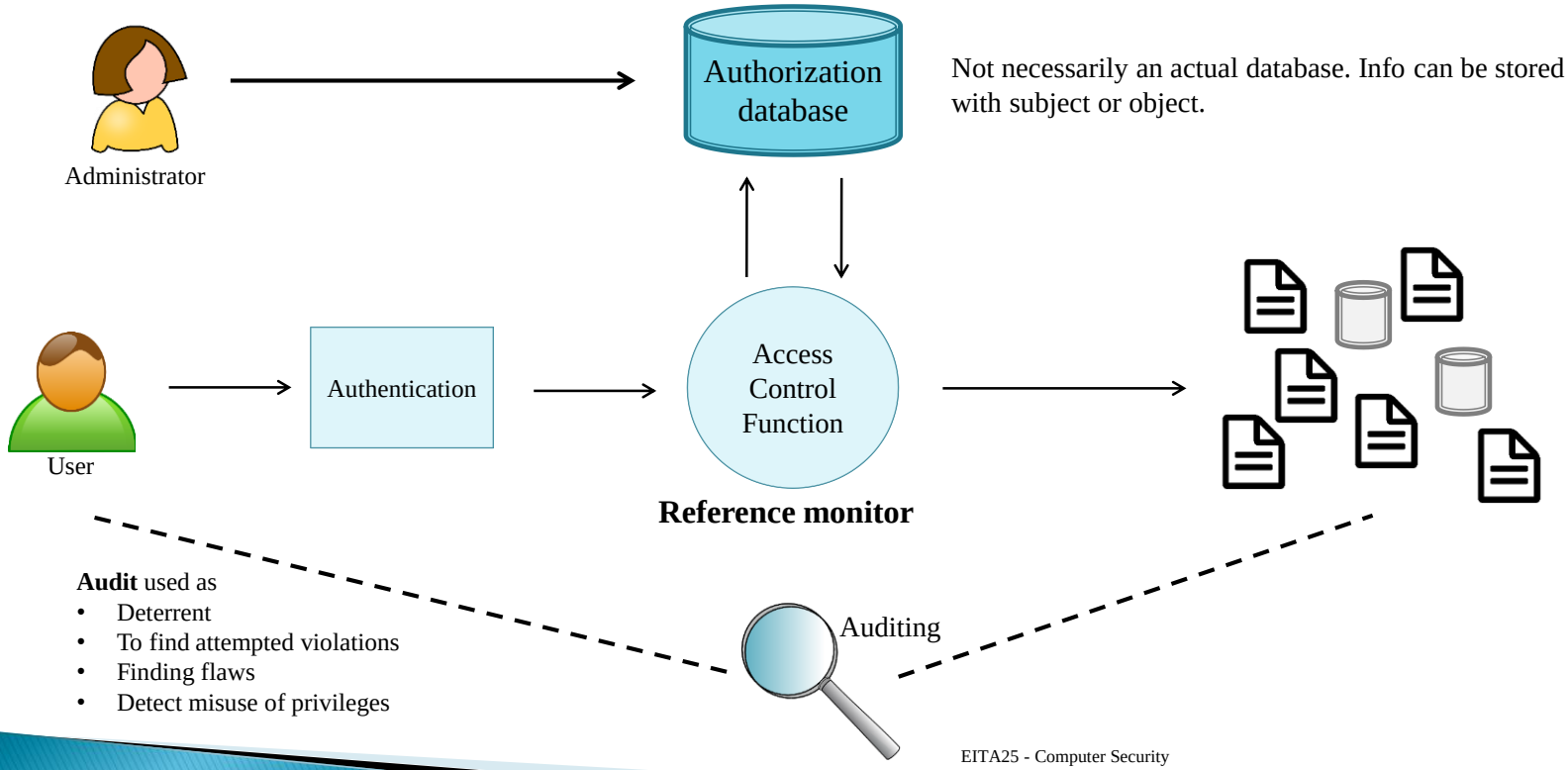
Goal of access control: Protection of system resources against unauthorized access.

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

Access Control Context



Two Important Principles

- Principle of least privilege (need-to-know principle)
  - 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 in 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

Subjects, Objects, and Access Rights

- Subject/Object
  - A *subject* is an entity capable of accessing objects
  - An *object* is a resource to which access is controlled
  - *NOTE:* In some litterature there is a distinction between subject and principal, where subject (process) acts on behalf of a principal (user, UID)
- 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 Rights

Elementary level:

- Observe: look at the contents of an object
  - Compare to confidentiality
- Alter: change the contents of an object
  - Compare to integrity
- This is often too general to be practical
- Applications or operating systems define other access right
  - Read
  - Write
  - Execute
  - Delete
  - Create
  - Search

Detailed meaning is defined by application/OS

Example: Bell-LaPadula security model:
Execute, Append, Read, Write

	Execute	Append	Read	Write
Observe			X	X
Alter		X		X

Example: Unix (files)
Read, Write, Execute

	Execute	Read	Write
Observe		X	
Alter			X

Access Control Policy

- **Discretionary access control (DAC)** – Access is restricted based on the identity of the subject. The owner of an object can decide the access rights.
- **Mandatory access control (MAC)** – Access is restricted based on the information sensitivity of an object and the authorization level of a subject. The system decides the access rights.
- **Roll-based access control (RBAC)** – Access is restricted based roles that users have within the system
- **Attribute-based access control (ABAC)** – Access is restricted based attributes of users, resources and environment

Discretionary Access Control, Access Matrix

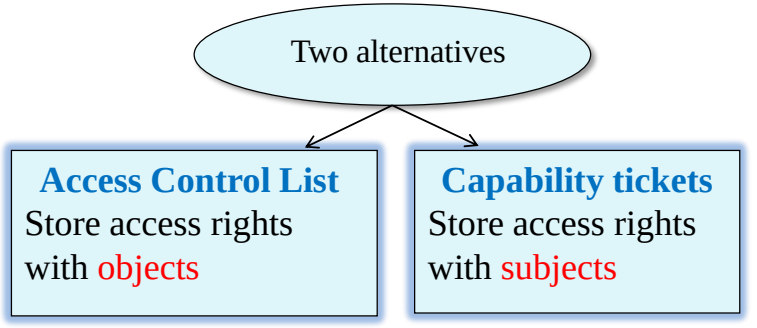
- Access rights individually defined for each subject and object
- Let
  - S: the set of subjects
  - O: the set of objects
  - A: the set of access operations
- The access rights are uniquely defined by the *access matrix* M, with $M_{so} \subseteq A$, $s \in S$, $o \in O$

Can also be groups of users →

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

Access 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



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

An ACL can include default access rights for unlisted users

- Follow the rule of least privilege

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

Capability Ticket

Separate each row

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

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

Tickets can be passed to other subjects. Integrity must be protected!

- Held by OS
- Include MAC

Difficult to determine who has access to a given object

Authorization Table

Capability list

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

Sort by subject

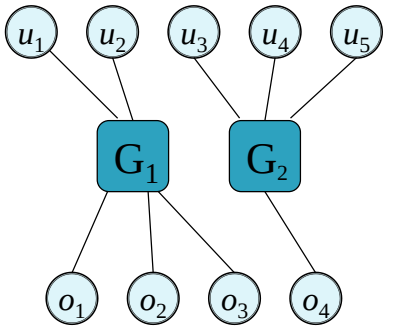
Access Control List

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

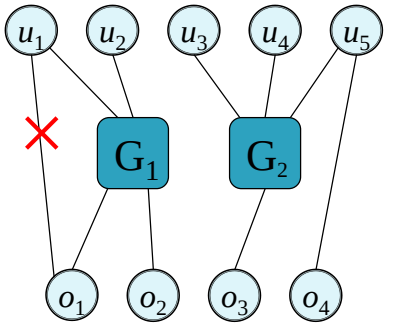
Sort by object

Groups and Negative Permissions

Putting users into a group can simplify



What about *policy conflicts!*



Should u_1 get access to o_1?

Denying access may not be the same as not allowing access

Protection Domains

A protection domain specifies which objects a user can access, and with which access rights

Protection domain associated with users

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

Generalization

Protection domain 1

O_1, {read, execute}
O_2, {read,write}
O_3; {write}

Protection domain 2

O_1, {read, write}
O_2, {read}
O_3, {write}

Now, each process can be associated with a protection domain (statically or dynamically)

- **Static:** Need a way to modify the protection domain
- **Dynamic:** Need a way to switch domain

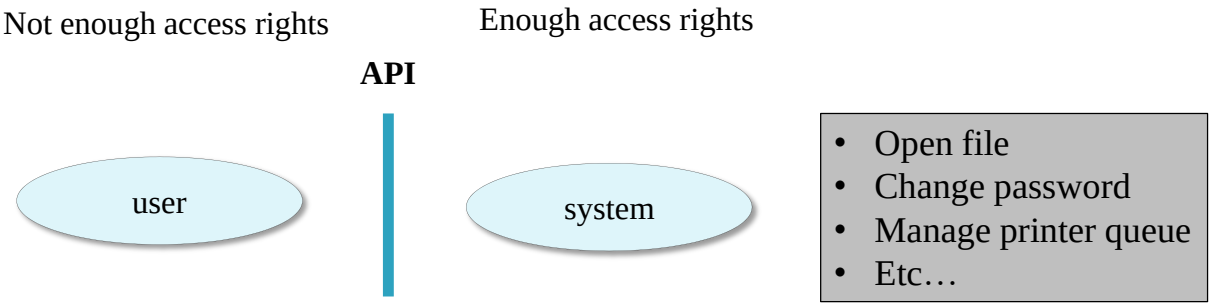
Possible to spawn processes with fewer access rights

Example 1
SetUID in Unix/Linux allows process to switch to a different domain

Example 2
Processors support user mode and kernel mode and some instructions are only allowed in kernel mode

Controlled Invocation

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



Mandatory Access Control

- Give each subject a clearance (authorization level)
- Give each object a classification (based on sensitivity)
- Access is based on clearance and classifications

} **Security labels**

Define relations between security labels in order to define policy and implement the access control. Relation is used to order the security labels.

- Policy and access rights are situation dependent
  - **Confidentiality**: Subject with low clearance cannot read objects with high classifications (but can write to them)
  - **Integrity**: Subject with low clearance cannot write objects with high classifications (but can read them)

} This will be formalized later in the course

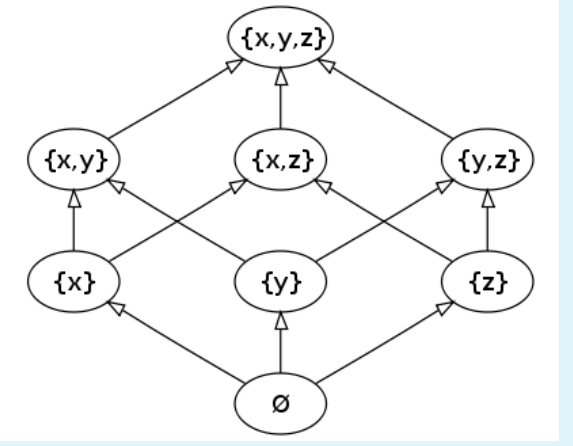
Relations Using Powerset

- 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} }

- Let ≤ denote the relation (subset). 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}

Visualization



There is an edge between node *a* and *b* if and only if

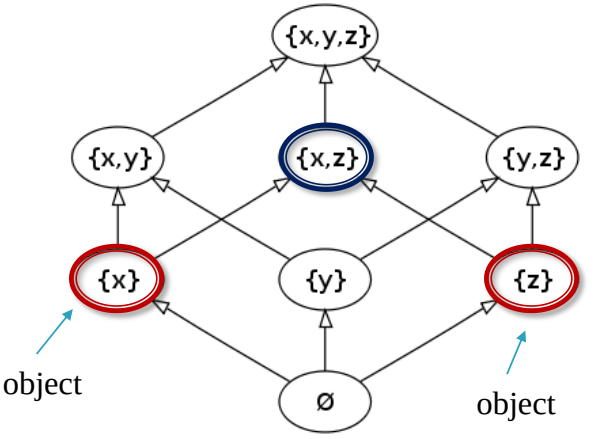
- a ≤ b and a ≠ b
- There is no c ∈ L so that a ≤ c ≤ b and a ≠ c, b ≠ c

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 relation $\leq$. 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 is a lattice

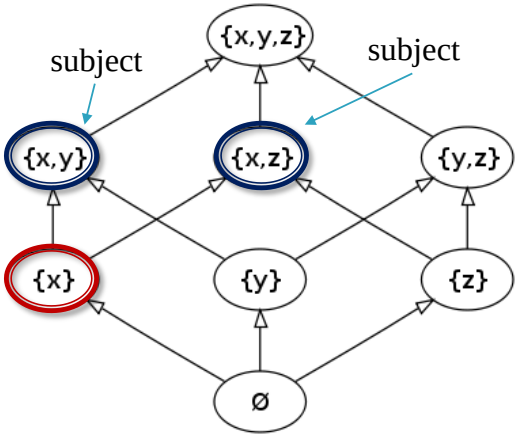
Subset Relation is a Lattice

- Example of least upper bound *u*
  - Let *a* = {x} and *b* = {z}
  - What is the least upper bound?
  - Then *u* = {x,z}
  - {x} ≤ {x,z} and {z} ≤ {x,z} and for all elements *v* such that *a* ≤ *v* and *b* ≤ *v* we also have *u* ≤ *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}

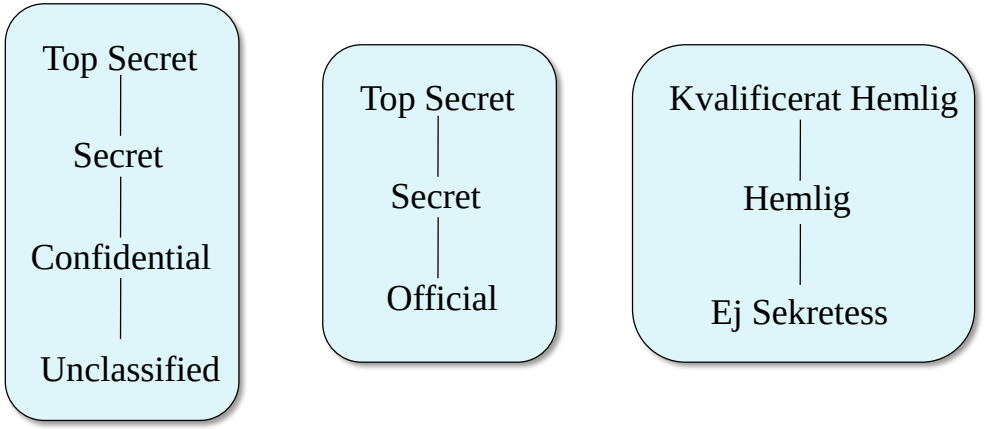


Subset Relation is a Lattice

- Example of greatest lower bound *l*
  - Let *a* = {x,y} and *b* = {x,z}
  - What is the greatest lower bound?
  - Then *l* = {x}
  - {x} ≤ {x,y} and {x} ≤ {x,z} and for all elements *k* such that *k* ≤ *a* and *k* ≤ *b* we also have *k* ≤ *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



U.S. U.K. Sweden

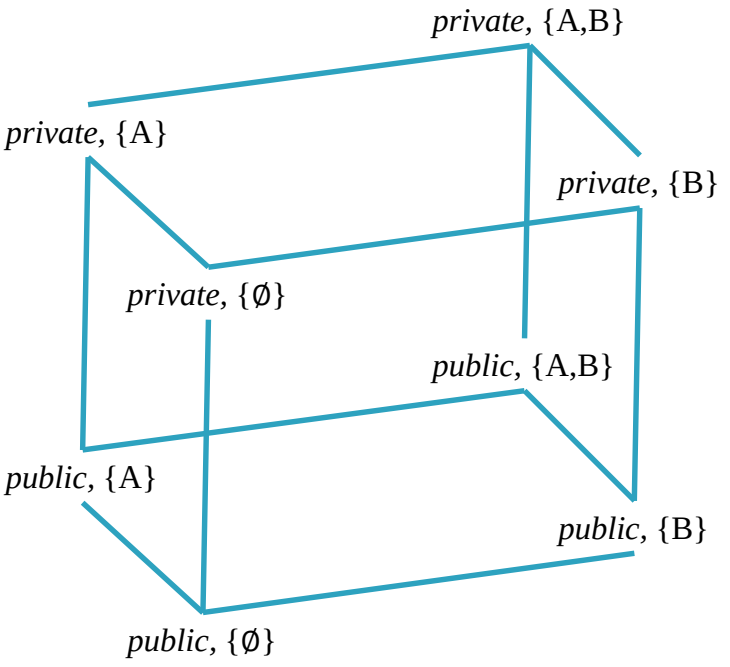
Combine with a set of categories to support a *need to know policy*:

Security level is (h,c), $h \in H$, $c \in C$
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

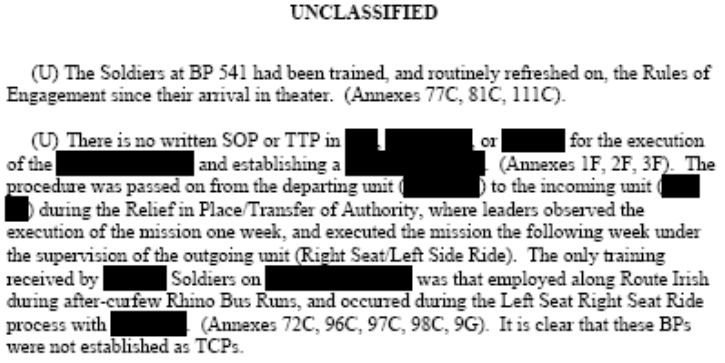
- Example of multilevel security with categories
  - 2 levels: *public* and *private*
  - 2 categories: *A* and *B*

Example relations:
(public, {A}) ≤ (private, {A})
(public, {B}) ≤ (public, {A,B})
(public, {B}) ⋦ (private, {A})



Sanitizing Information

Users with lower clearance must sometimes share their work with users of lower clearance



UNCLASSIFIED

(U) The Soldiers at BP 541 had been trained, and routinely refreshed on, the Rules of Engagement since their arrival in theater. (Annexes 77C, 81C, 111C).

(U) There is no written SOP or TTP in [redacted] or [redacted] for the execution of the [redacted] and establishing a [redacted]. (Annexes 1F, 2F, 3F). The procedure was passed on from the departing unit ([redacted]) to the incoming unit ([redacted]) during the Relief in Place/Transfer of Authority, where leaders observed the execution of the mission one week, and executed the mission the following week under the supervision of the outgoing unit (Right Seat/Left Side Ride). The only training received by [redacted] Soldiers on [redacted] was that employed along Route Irish during after-curfew Rhino Bus Runs, and occurred during the Left Seat Right Seat Ride process with [redacted] (Annexes 72C, 96C, 97C, 98C, 9G). It is clear that these BPs were not established as TCPs.

Role-based Access Control (RBAC)

- Access rights are derived from a user's current role – not identity
- **Example:** User + current job → role
- Motivation:
  - Users come and go
  - Users have different roles at different times
  - Roles are often more static
  - Roles' access rights to resources are more static
- Principle of least priviledge
  - Each role has minimum right needed to perform its task
  - Users can have many roles

	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		

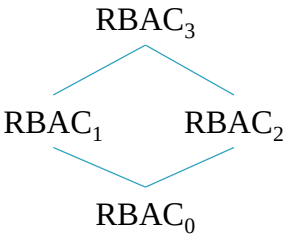
m ≫ n

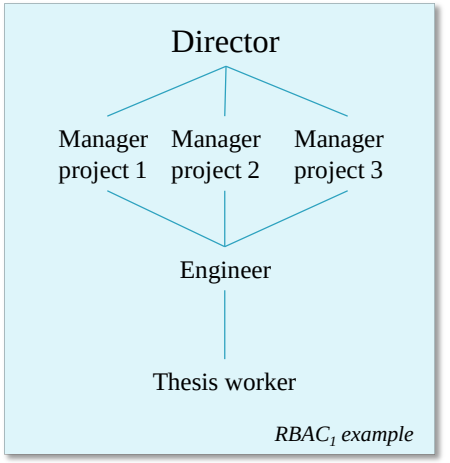
	Object 1	Object 2	Object 3		Object k
Role 1	owner	modify	stop, start		copy
Role 2	append				
Role 3	read		start		
Role n	read				defrag

k ≫ n

RBAC Reference 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 by allowing access to be limited to subordinate roles)
- **$RBAC_3$**: $RBAC_1$ + $RBAC_2$
  - Combining hierarchies with constraints





Attribute-Based Access Control

- Base access on properties (attributes) of both subject, resource and environment
- Key elements
  - Attributes
  - Architectural model
  - Policies
- Very powerful and support arbitrarily fine grained access control
  - Resource consuming – possible performance impact

Attributes

Three types

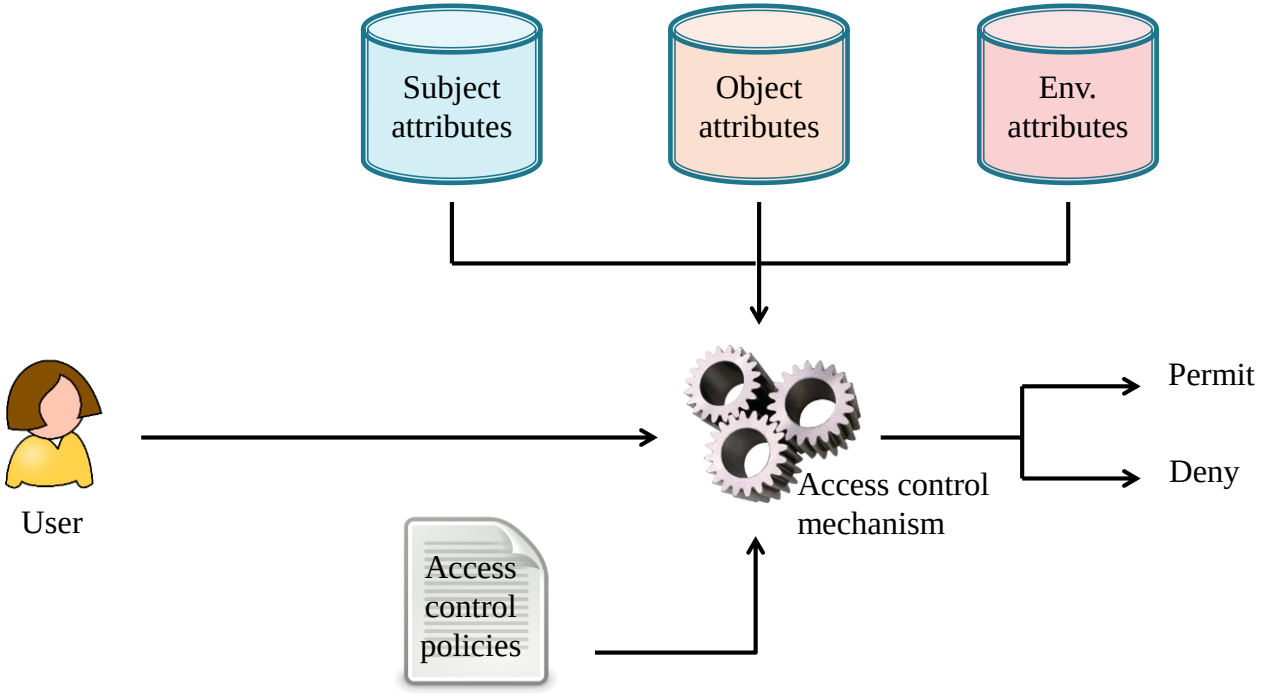
Subject attributes. Define identity and characteristics of the subject
Examples: Name, organization, job title, division, age

Object (resource) attributes. Define characteristics of the object
Examples: Document title, author, subject, creation time, associated people

Environment attributes. Describe context in which information access occurs
Examples: Date and time, country of access, system properties

Note: ABAC is very general and can be used to enforce DAC, MAC and RBAC

Architectural Model



Policies

- A set of rules that define allowable behavior

Possible policy model

1. Let *S*, *O* and *E* be subjects, objects and environment
2. SA_k ($1 \leq k \leq K$), OA_m ($1 \leq m \leq M$) and EA_n ($1 \leq n \leq N$) are the attributes for subjects, objects and environment
3. ATTR(s), ATTR(e), and ATTR(o) are the assignments of attributes

$$\begin{aligned} \mathrm{ATTR}(s) &\subseteq \mathrm{SA}_1 \times \mathrm{SA}_2 \times \ldots \times \mathrm{SA}_K \\ \mathrm{ATTR}(o) &\subseteq \mathrm{OA}_1 \times \mathrm{OA}_2 \times \ldots \times \mathrm{OA}_M \\ \mathrm{ATTR}(e) &\subseteq \mathrm{EA}_1 \times \mathrm{EA}_2 \times \ldots \times \mathrm{EA}_N \end{aligned}$$

4. Policy rule is a Boolean function taking attributes as input

$$\text{Rule: } \mathrm{can\_access(s, o, e)} \leftarrow f(\mathrm{ATTR}(s), \mathrm{ATTR}(o), \mathrm{ATTR}(e))$$

5. Policy rule base or policy store is a set of rules that are evaluated

Example, RBAC vs. ABAC

Online movie streaming website

Policy

Movie Rating	Allowed to stream
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

RBAC

Manually map user-to-role and permission-to-role

Reasonable roles

	Adult	Juvenile	Child
User 1	X		
User 2		X	
User 3	X		
User 4			X
User 5		X	
:			

	Bamse	Star Wars	The Shining	Batman	…
Adult	X	X	X	X	
Juvenile	X	X		X	
Child	X				

ABAC

Use actual age (subject attribute) and movie rating (object attribute)

R1: can_access(u, m, e) ←

$$(\text{Age}(u) \geq 17 \wedge \text{Rating}(m) \in \{\text{R}, \text{PG-13}, \text{G}\}) \vee$$
$$(\text{Age}(u) \geq 13 \wedge \text{Age}(u) < 17 \wedge \text{Rating}(m) \in \{\text{PG-13}, \text{G}\}) \vee$$
$$(\text{Age}(u) < 13 \wedge \text{Rating}(m) \in \{\text{G}\})$$

What if we update the policy as
"Premium users can watch new releases and old releases, while regular users can only watch old releases"?

Example, RBAC vs. ABAC

Online movie streaming website

Policy

Movie Rating	Allowed to stream
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone
New Release	Premium users
Old Release	All users

P = Premium
R = Regular

RBAC

Manually map user-to-role and permission-to-role

	Adult/P	Adult/R	Juvenile/P	Juvenile/R	Child/P	Child/R
User 1	X					
User 2				X		
User 3		X				
User 4					X	
User 5			X			
:						

	Bamse	Star Wars	The Shining	Batman	Sune vs. Sune	Avengers 4	The Ring 5
Adult/P	X	X	X	X	X	X	X
Adult/R	X	X	X	X			
Juvenile/P	X	X		X	X	X	
Juvenile/R	X	X		X			
Child/P	X				X		
Child/R	X						

Example, RBAC vs. ABAC

Each user is related to a role

```
{Adult,     Premium}
{Adult,     Regular}
{Juvenile,  Premium}
{Juvenile,  Regular}
{Child,     Premium}
{Child,     Regular}
```

$$\begin{aligned} \mathrm{SA}_1 &= \{\text{Adult}, \text{Juvenile}, \text{Child}\} \\ \mathrm{SA}_2 &= \{\text{Premium}, \text{Regular}\} \end{aligned}$$

Number of Roles

$$\prod_{k=1}^{K} \mathrm{Range}(\mathrm{SA}_k) = 6$$

Each movie is related to a permission

```
{R,     New Release}
{R,     Old Release}
{PG-13, New Release}
{PG-13, Old Release}
{G,     New Release}
{G,     Old Release}
```

$$\begin{aligned} \mathrm{OA}_1 &= \{\text{R}, \text{PG-13}, \text{G}\} \\ \mathrm{OA}_2 &= \{\text{New Release}, \text{Old Release}\} \end{aligned}$$

Number of permissions

$$\prod_{m=1}^{M} \mathrm{Range}(\mathrm{OA}_m) = 6$$

Example, RBAC vs. ABAC

Online movie streaming website

Policy

Movie Rating	Allowed to stream
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone
New Release	Premium users
Old Release	All users

ABAC

Use actual age (subject attribute), membership type (subject attribute), movie rating (object attribute) and movie type (object attribute)

R1: can_access(u, m, e) $\leftarrow$

$$(\text{Age}(u) \geq 17 \wedge \text{Rating}(m) \in \{\text{R}, \text{PG-13}, \text{G}\}) \vee$$
$$(\text{Age}(u) \geq 13 \wedge \text{Age}(u) < 17 \wedge \text{Rating}(m) \in \{\text{PG-13}, \text{G}\}) \vee$$
$$(\text{Age}(u) < 13 \wedge \text{Rating}(m) \in \{\text{G}\})$$

R2: can_access(u, m, e) $\leftarrow$

$$(\text{MembershipType}(u) = \text{Premium}) \vee$$
$$(\text{MembershipType}(u) = \text{Regular} \wedge \text{MovieType}(m) = \text{OldRelease})$$

R3: can_access(u, m, e) $\leftarrow R1 \wedge R2$

Easy to increase flexibility