CS458: Introduction to Information Security

Notes 11: Access Control

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

April 17th 2024

Slides: Adopted/Modified from Steven Gordon, Computer Security: Principles and Practice, 4th Edition. By: William Stallings and Lawrie Brown, & Wenliang (Kevin) Du, Syracuse University

Outline

- Access Control Concepts
- Discretionary Access Control (DAC)
- Role-Based Access Control (RBAC)
- Mandatory Access Control (MAC)
- Attribute-based access control (ABAC)

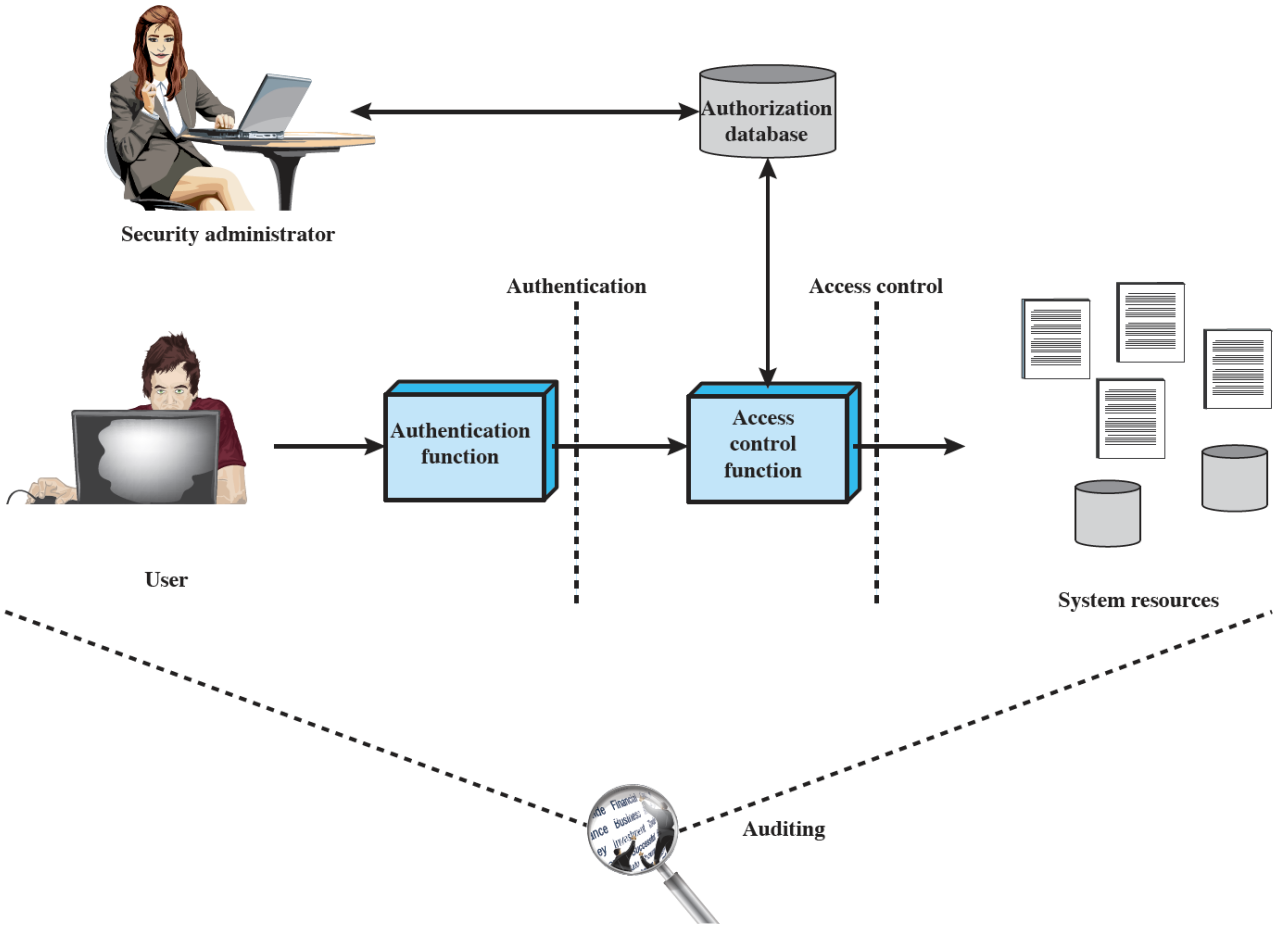
Access Control Definitions

- Three definitions of access control are useful in understanding its scope.
  - "the process of granting or denying specific requests to: (1) obtain and use information and related information processing services; and (2) enter specific physical facilities"[1]
  - "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"[2]
- "The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner"[3]
- Access control implements a security policy that specifies who or what (e.g., in the case of a process) may have access to each specific system resource and the type of access that is permitted in each instance.
- We can view access control as the central element of computer security.

[1] NISTIR 7298 (Glossary of Key Information Security Terms , May 2013)
[2] RFC 4949, Internet Security Glossary
[3] ITU-T Recommendation X.800 "Security architecture for Open Systems Interconnection"

Relationship Among Access Control and Other Security Functions

With respect to user authentication, access control is another function involved in our computer system.



Access Control and Other Security Functions

- Access control part of a broader context
  - Authentication - Verification that the credentials of a user or other entity are valid. Are you who you say you are? Who goes there?
    - Determine whether access is allowed
    - Authenticate human to machine
    - Authenticate machine to machine
  - Authorization - The granting of a right or permission to a system entity to access a resource. Are you allowed to do that?
    - Follows authentication.
    - Once you have access, what can you do?
    - Enforces limits on actions. Determine who is trusted for a given purpose.
  - A security administrator maintains an authorization database that specifies what type of access to which resources is allowed for each user.
  - The access control function consults this database to determine whether to grant access.
    - "Access control" often used as synonym for authorization.
    - The verification of access rights is access control, and the granting of access rights is authorization. These two terms are often used interchangeably
  - Audit
    - Independent review and examination of system records and activities in order to test for adequacy of system control, ensure compliance to policy, detect breaches in security and recommend changes

Access Control Policies

- Access control policies are fundamental components of information security, specifying the rules and conditions under which access to resources is granted or denied.
  - dictates what types of access are permitted, under what circumstances, and by whom.
- These policies are often implemented through authorization databases, where rules and permissions are stored.
- There are several types of access control policies, each with its own principles and characteristics.
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  - Role-based Access Control (RBAC)
  - Attribute-based access control (ABAC)

Access Control Policies

- Discretionary Access Control (DAC)
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
  - Regular users can adjust the policy, i.e., entities may allow other entities to access resources
- Mandatory Access Control (MAC)
  - Controls access based on comparing security labels with security clearances
    - *Security labels: indicate how sensitive or critical system resources are*
    - *Security clearances: indicate system entities are eligible to access certain resources.*
  - Regular user can not adjust the policy, i.e., entities cannot grant access to resources to other entities
- Role-based Access Control (RBAC)
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles
    - i.e., roles of users in system and rules for roles are used to control access
- Attribute-based access control (ABAC)
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

- DAC, MAC, RBAC, and ABAC are not mutually exclusive

Basic Elements of Access Control System

- Subject - entity capable of access resources (objects)
  - Often subject is a software process (often run by human users)
  - Classes of subject, e.g., Owner, Group, World.
- Object - resource in a system to which access is controlled
  - e.g. records, blocks, pages, files, portions of files, directories, email boxes, programs, communication ports
  - i.e., entity used to contain and/or receive information
- Access right - describe a way in which a subject may access an object
  - e.g., read, write, execute, delete, create, search

General Requirements of Access Control

- Reliable input
  - Access control assumes user has been authenticated, that other control inputs (e.g., object names, addresses) are correct
  - i.e., we assume that the authentication is reliable and it works.
- Fine and coarse specifications
  - We should be allowed to be very fine detailed in specifying who can access what, and also allowed very coarse specification of who can access what.
  - Striking a balance between management overhead and policy precision is crucial to ensure effective access control.
- Least privilege
  - This principle dictates users or entities should be provided with minimum set of privileges or permissions necessary to accomplish their tasks.
  - This minimizes the potential impact of security breaches, as users only have access to what is essential for their roles or responsibilities.
- Separation of duty
  - A security principle aims to prevent fraud and errors by distributing tasks and associated privileges among multiple users.
  - This involves ensuring that no single individual has complete control over a critical process or system.
  - By requiring the collaboration of multiple entities to complete a task, separation of duty increases protection against fraud and errors.

General Requirements of Access Control

- Open and closed policies: refer to the approach taken in defining access permissions.
  - "closed": access limited to those explicitly stated
    - e.g., default "deny" on firewall rule
  - "open": access limitations are specified, all others allowed
- Policy combinations and conflict resolution
  - In real-world scenarios, different entities may have different access control policies.
  - When combining these policies, conflicts can arise.
  - Conflict resolution mechanisms are necessary to adjudicate between conflicting policies and ensure that access control decisions are consistent and unambiguous.
- Administrative policies
  - Need to allow admin user to the system
- Dual control
  - In some cases, have multiple people/users to access the same resources in parallel at the same time or it requires two or more people to perform a function
  - Two entities required to implement policy change (e.g., should allow to have multiple admins)

Discretionary Access Control (DAC)

- DAC is a common access control scheme used in operating systems and database management systems.
- It provides a flexible approach to managing access permissions, especially in scenarios where users need the flexibility to control access to their own resources.
- In DAC, an entity (user, process, or system) is given access rights to specific resources.
- Importantly, the entity has the discretion to choose whether or not to enable another entity to access those resources. This delegation of access rights is at the discretion of the original entity.
- Often provided using an access matrix (Access control matrix) which specifies access rights of subjects on objects.
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

Example of Access Control Matrix

- A matrix with each subject represented by a row, and each object represented by a column
- The entry `M[s,o]` lists the operations that subject `s` may carry out on object `o`
- Subjects (users) index the rows.
- Objects (resources) index the columns

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

r - read, w - write, x - execute

Are you allowed to do that?

- Access control matrix has all relevant information to answer this question.
  - Could be `100's` of users, `10,000's` of resources.
    - Then matrix with `1,000,000's` of entries.
  - How to manage such a large matrix?
    - We need to check this matrix before access to any resource by any user is allowed.
  - Is this matrix a good way to represent access rights?
  - How to make this efficient/practical?

- *A matrix with hundreds of users and thousands of resources can result in millions of entries, making it challenging to manage and evaluate efficiently.*

Implementing an Access Matrix

- There are two main approaches that are used instead of an actual matrix:
  - Split it into smaller pieces: by rows or by columns
- Access Control Lists (ACL): *associated with resources*
  - Each object can maintain a list, the access control list, of the access rights of subjects that want to access that object - this effectively distributes the matrix column-wise, leaving out empty entries
    - *For each object, list subjects and their access rights*
- Capability Lists (Capability tickets)
  - Each subject can maintain a list of capabilities for objects - this effectively distributes the matrix row-wise, leaving out empty entries
    - *Capabilities are tokens or keys that users possess, allowing them to access specific resources directly.*
    - *For each subject, list objects and the rights the subject have on that object*
  - Of course, capabilities can't be totally maintained by the subjects - they must be given to the subjects by some other trusted entity
- Alternative implementation: authorization table listing subject, access mode and object; easily implemented in database

Access Control Lists (ACLs)

- ACL stores access control matrix by column (Slice by column)
  - ACL for insurance data in blue.

	OS	Accounting program	Accounting data	**Insurance data**	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

r - read, w - write, x - execute

- ACL may contain a default, or public, entry. This allows users that are not explicitly listed as having special rights to have a default set of rights.
- The default set of rights should always follow the rule of least privilege or read-only access, whichever is applicable.

Capabilities (or C-Lists)

- Store access control matrix by row (Slice by row)
  - Capability for Alice in red

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

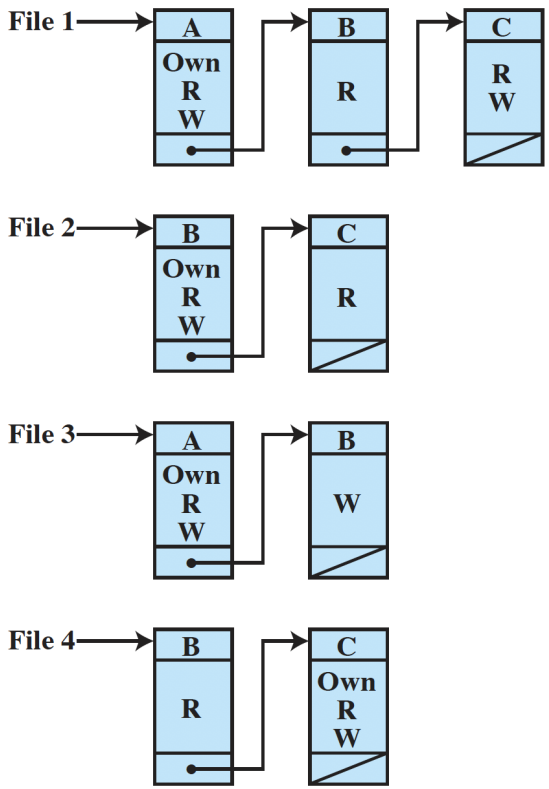
r - read, w - write, x - execute

Example of Access Control Structures

		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

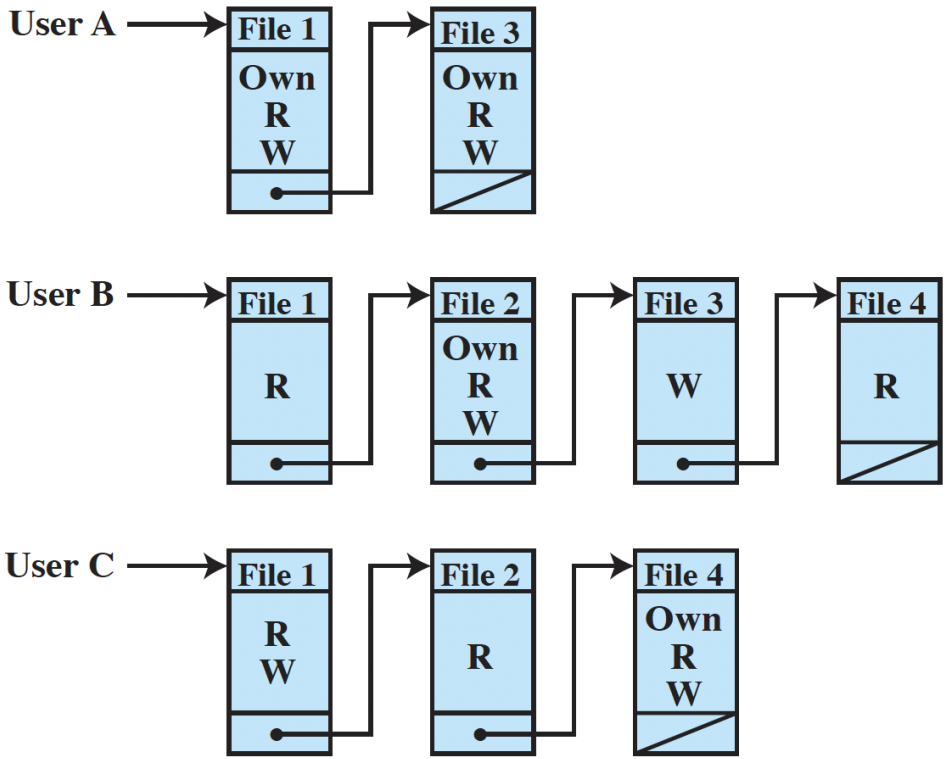
- `User A` owns `files 1` and `3` and has `read` and `write` access rights to those files.
- `User B` has `read` access rights to `file 1`, and so on.

Example of Access Control Lists



- ACLs are convenient, when it is desired to determine which subjects have which access rights to a particular resource
- Not convenient for determining the access rights available to a specific user

Example of Capability Lists



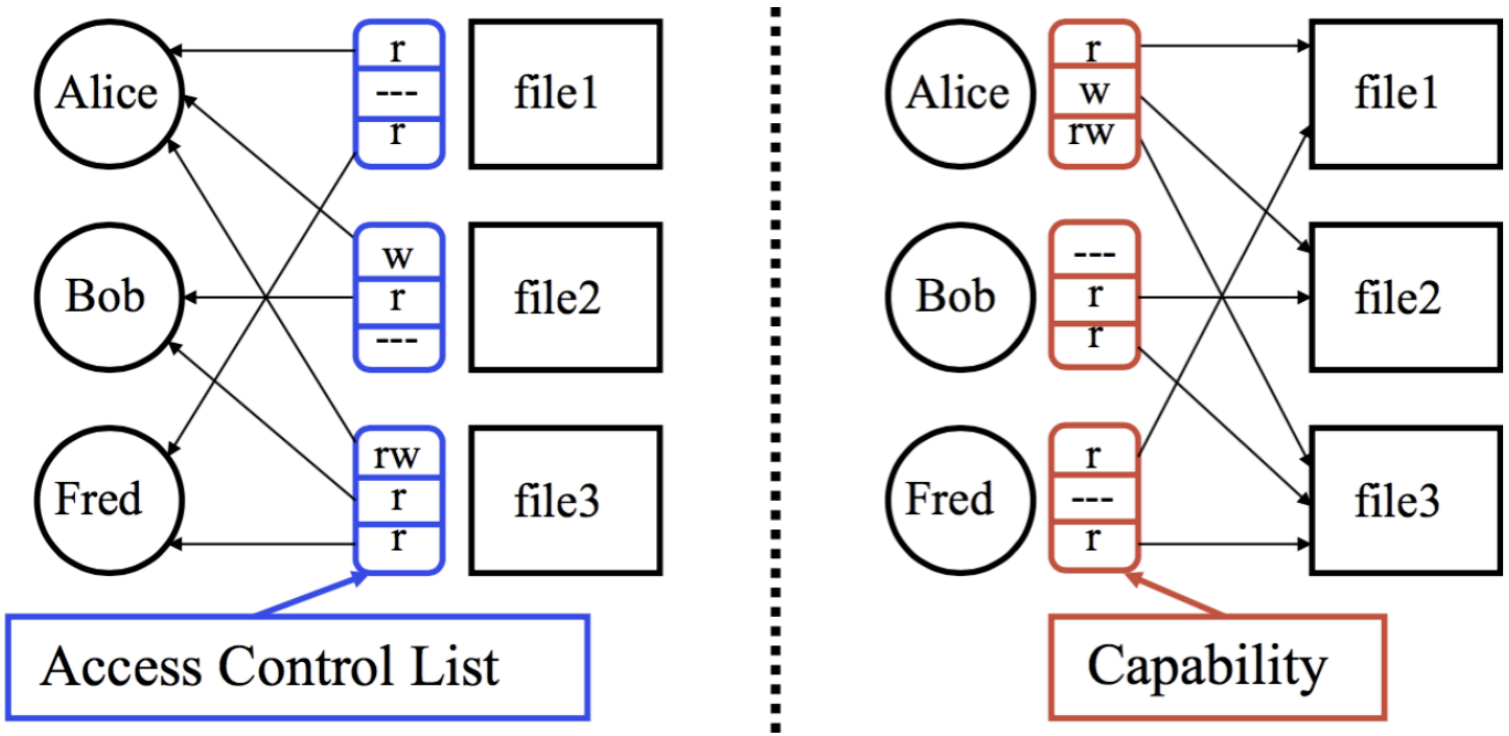
Capabilities Integrity

- Decomposition by rows yields capability tickets
- A capability ticket specifies authorized objects and operations for a particular subject
- Subject presents "capability" in order to access object
  - Capability data structure encapsulates object ID with allowed rights.
- Each user/subject has a number of tickets and may be authorized to loan or give them to others
- Unlike ACLs, capabilities are not entirely contained within the OS
  - They can be transmitted, shared, or even loaned to other subjects. This flexibility allows for dynamic collaboration but introduces security challenges.

Capabilities Integrity

- Because capabilities can be dispersed throughout the system and transferred between subjects, they present a greater security challenge compared to centralized access control models like ACLs.
- The integrity of capability tickets is a significant concern.
  - To address integrity concerns, the operating system typically holds all capability tickets on behalf of users.
    - These tickets may be stored in a region of memory that is inaccessible to users, ensuring that they cannot tamper with or forge their own capabilities.
  - Guaranteeing the unforgeability of capability tickets is essential.
    - This is often achieved by including an unforgeable token within the capability data structure.
    - This token can be a random password or a cryptographic message authentication code (MAC) that proves the authenticity of the capability.

ACLs vs Capabilities



- Note that arrows point in opposite directions.
- ACLs: Given an object, which subjects can access it?
- Capabilities: Given a subject, what objects can it access?

Authorization Table

- Data structure not sparse, like the access matrix
  - Each row describes one access right of one subject to one resource
  - Sorting or accessing the table by subject is equivalent to a capability list
  - Sorting or accessing the table by object is equivalent to an ACL
  - Easy implemented with relational database

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

Protection Domains

- The access control matrix model that we have discussed so far associates a set of capabilities with a user.
- A more general and more flexible approach proposed is to associate capabilities with protection domains
- A protection domain is a set of (object, access rights) pairs, where each pair specifies for a given object exactly what operations can be carried out
  - i.e., a set of objects together with access rights to those objects
- Associating capabilities with protection domains provides greater flexibility.
- In Access Matrix, replace user with "Protection Domain".
- In this model, each user is associated with a protection domain.
- At runtime, the process representing a user may run under different protection domains, depending on the specific context or task.

Protection Domains

- The concept of protection domains aligns with the idea of *user mode* and *kernel mode* in computing systems.
  - In *user mode*, certain areas of memory are protected from use, and certain instructions may not be executed.
    - Users operate within the constraints of this mode, limiting their access to privileged operations.
  - In *kernel mode*, privileged instructions may be executed, and protected areas of memory may be accessed.
    - The kernel has elevated privileges and can perform critical system-level operations.

- The ability for a process to run under different protection domains at runtime adds a layer of flexibility.
  - This allows for the adjustment of access rights and permissions based on specific conditions or requirements.

UNIX File Access Control (FAC)

- FAC is used in most UNIX-like operating systems, including Linux, macOS, and FreeBSD.
- It is a fundamental mechanism for controlling access to files and directories, and it is essential for maintaining security and privacy in multi-user environments.
- The FAC system is based on the concept of inodes (index nodes), which are data structures that store information about files and directories.
- Each inode has a unique identifier, and it contains information such as the file's owner, permissions, size, and timestamps.
- The inode also contains pointers to the data blocks that store the actual contents of the file.
    - mode
    - owner information
    - size
    - timestamps
    - pointers to data blocks (data blocks contain the actual file)

UNIX File Access Control (FAC)

- The OS maintains a list of inodes in inode table that contains the inodes of all the files in the file system.
- When a file is opened, its inode is brought into main memory and stored in a memory-resident inode table.
  - This allows the operating system to quickly access the file's information and data.
- A directory is a special type of file that lists entries for each file and directory in that directory.
  - Each directory entry contains the inode number of the file or directory, the length of the file name, and the file name.
    - inode number of file
    - length of name of file
    - name of file

inode Contents

- mode 16 bits
  - `12` protection bits: permissions
  - `4` bit file type: regular file, directory, ...
- owner id 16 bit user ID
- group id 16 bit group ID
- size size of file in bytes
- timestamps last time, in seconds since epoch:
  - `atime`: inode accessed
    - access time: updated when the file's contents are read by application or a command such as `grep` or `cat`.
  - `ctime`: inode changed
    - change time: when the file's property changes (changes when the `mtime` changes, you change file's permissions, name or location)
    - will update the `atime`
  - `mtime`: file data modifies
    - modification time: when the file was last modified (you change the contents of the file)
    - will update both `atime` and `ctime`

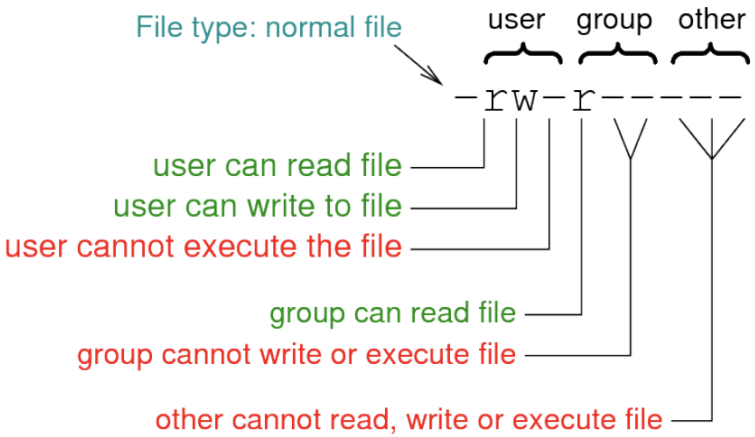
and other fields ...

Permissions and Users

- Permissions
  - `read` the file; list the contents of the directory
  - `write` the file; create and move files in the directory
  - `execute` the file; access file in the directory
- Categories of Users
  - `user` that owns the file
  - users in the file's `group`
  - `other` users
  - (`all` users, i.e., the above three)

Protection bits in an inode

- `12` bits in an inode are protection bits
  - First `9` bits indicate **`read`**, **`write`**, **`execute`** permissions for user, group and others.
  - Last `3` bits indicate special permissions
- File type (regular or directory) and values of protection bits shown in user-friendly format
  - First letter indicates file type: **`d`**irectory; **`-`** is normal file
  - Next `9`:
    - Letter indicates the permission is set;
    - **`-`** indicates thee permission is not set



Role-Based Access Control (RBAC)

- DAC systems define the access rights of individual users and groups of users.
- RBAC is based on the roles that users assume in a system rather than the user's identity
- Roles typically job functions or positions within an organization, e.g. senior financial analyst in a bank, doctor in a hospital
- A user has access to an object based on the assigned role.
- RBAC systems assign access rights to roles instead of individual users
  - i.e., users are assigned to roles; access rights are assigned to roles
- Each role will have specific access rights to one or more resources
- Users may be assigned multiple roles; static or dynamic according to their responsibilities.
- Sessions are temporary assignments of user to role(s)
- Access control matrix can map users to roles and roles to objects

DAC vs RBAC

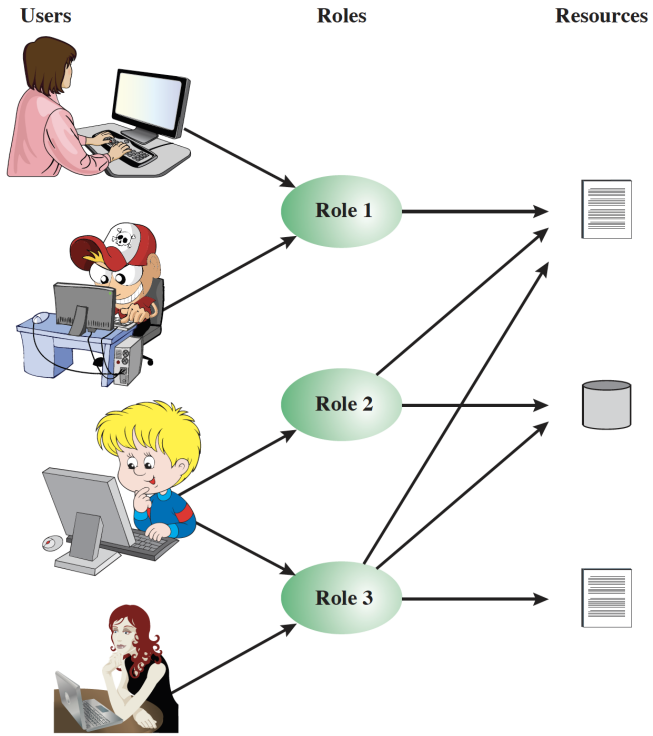
- DAC
  - Users, Groups → Permissions
- RBAC
  - Roles → Permissions
  - Users → Roles
  - Many-to-many relations
- Difference between DAC and RBAC?
  - Different perspectives:
    - RBAC is from perspective of organization
  - Different right management:
    - DAC: allow a user to grant access to the objects he/she owns
    - RBAC: users cannot pass their rights to others
- *Groups are primarily collections of users with associated permissions, while roles are collections of permissions that users can assume under specific conditions. Roles offer more fine-grained control over permissions and are often used in situations where access needs are more dynamic or context-dependent. Both groups and roles play essential roles in access control strategies, with groups providing a straightforward way to manage permissions for sets of users and roles offering a more sophisticated and context-aware approach to permission management.*

DAC vs RBAC

- Difference between groups and roles?
  - Group
    - A group is a collection of users who share a common characteristics or responsibilities.
    - Groups are primarily used to simplify permission management by assigning a set of permissions to the group, which are then inherited by all members of the group.
    - This approach is particularly useful when managing access to resources that are shared by a group of users with similar needs.
  - Role
    - A role defines a set of permissions that a user can assume under specific circumstances.
    - Roles are typically associated with specific tasks or activities within a system.
    - When a user assumes a role, they are temporarily granted the permissions associated with that role.
    - This allows users to have different access levels based on the context of their actions.

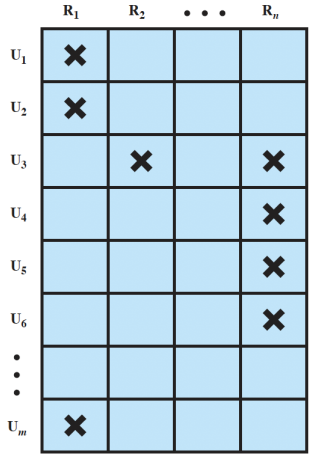
Role-Based Access Control (RBAC): Example

- The relationship of users to roles is many to many, as is the relationship of roles to resources, or system objects
  - One user may have multiple roles, and multiple users may be assigned to a single role.
- A session is used to define a temporary one-to-many relationship between the user and one or more of the roles to which the user has been assigned.
  - The user establishes a session with only the roles needed for a particular task (an example of the concept of least privilege)



Access Control Matrix Representation of RBAC

- We can use access matrix representation to depict the key elements of an RBAC system in simple terms



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			

- Upper matrix relates individual users to roles (one user per role)
  - e.g., role 1 has 3 users, user 3 can take 2 roles.
- Lower matrix has the same structure as the DAC access control matrix, with roles as subjects
- Each role should contain the minimum set of access rights needed for that role

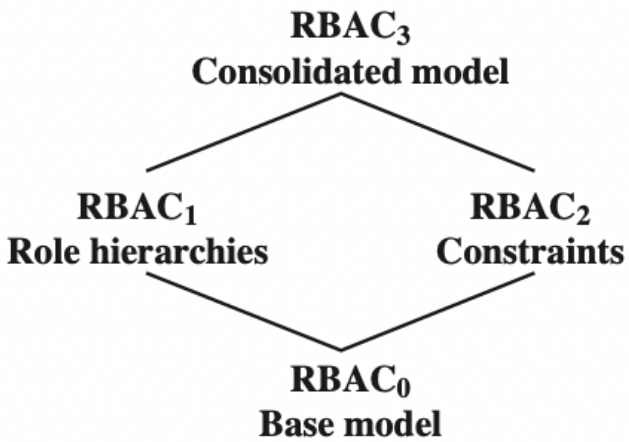
Role-Based Access Control

- RBAC is a rich and open-ended technology that can be used to implement simple or complex access control policies depending on the needs of the organization.
- Because RBAC can be used in a variety of different contexts and environments, treating RBAC as a single model is therefore unrealistic.
  - A single (one-size-fits-all) model would either include or exclude too much, and would only represent one point along a spectrum of technology and choices.
- A variety of functions and services can be included under the general RBAC approach.
- To better understand and implement RBAC, it is beneficial to define abstract models that capture its core functionalities.
  - These abstract models serve as conceptual frameworks for defining roles, permissions, and access control policies.

- *This flexibility ensures that RBAC can accommodate different levels of complexity and functionality.*

A Family of RBAC Reference Models

- SAND96 defines a family of reference models that has served as basis for ongoing standardization effort.
- Consists of four models that provide a comprehensive framework for understanding and implementing RBAC systems of varying complexity.
  - $\texttt{RBAC}_0$: Contains the minimum functionality for an RBAC system
  - $\texttt{RBAC}_1$: $\texttt{RBAC}_0$ `+ Role hierarchies`
    - enable one role to inherit permissions from another role.
  - $\texttt{RBAC}_2$: $\texttt{RBAC}_0$ `+ Constraints`
    - restrict the ways in which the components of a `RBAC` system may be configured
  - $\texttt{RBAC}_3$: $\texttt{RBAC}_0 + \texttt{RBAC}_1 + \texttt{RBAC}_2$



Relationship among RBAC models

Scope RBAC Models

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

Base Model: $\texttt{RBAC}_0$: The Core Model

- The core model of the SAND96 family, defines the basic concepts and relationships required for an RBAC system. It includes the following elements:
  - Users: Individuals who interact with the system.
  - Role: A named job function within the organization that controls the system
  - Permission: An approval of a particular mode of access to one or more objects.
    - Equivalent terms are access right, privilege, and authorization.
  - Session: A mapping between a user and a subset of roles
    - allows selective activation and deactivation of roles assigned

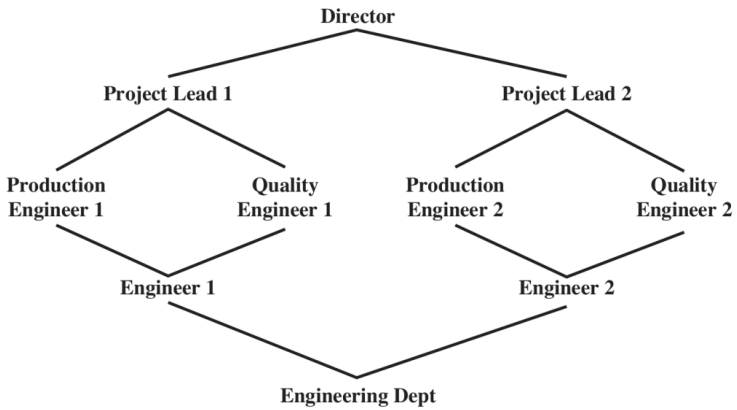
  - $\texttt{RBAC}_0$ establishes the fundamental principle of RBAC:
    - Users are assigned roles, and roles are assigned permissions.
    - This approach simplifies permission management and enables granular control over user access.

Session

- A user can invoke multiple sessions
- In each session a user can invoke any subset of roles that the user is a member of

Role Hierarchies: $\texttt{RBAC}_1$

- $\texttt{RBAC}_1$ extends $\texttt{RBAC}_0$ by introducing role hierarchies, which allow roles to inherit permissions from other roles.
- This creates a hierarchical structure of roles, enabling more efficient permission management and reflecting organizational structures.
- A role hierarchy is a tree-like structure in which roles are organized into levels, with higher-level roles having more permissions than lower-level roles.
- Roles at higher levels in the hierarchy can inherit permissions from roles at lower levels, providing a more flexible and scalable approach to access control.



Example of Role Hierarchy

Constraints: $\texttt{RBAC}_2$

- $\texttt{RBAC}_2$ includes $\texttt{RBAC}_0$ and adds constraints, which restrict the ways in which the components of a RBAC system may be configured.
- Constraints can be used to enforce policies such as separation of duty or least privilege, which help to prevent unauthorized access and reduce the risk of security breaches.
- Constraints define relationships between roles or conditions on roles.
- Types of Constraints:
  - Mutually exclusive roles:
    - A user can only be assigned to one role in the set (either during a session or statically)
    - Set of mutually exclusive roles have non-overlapping permissions. Any permission can be granted to only one role in the set
  - Cardinality: Setting a maximum number with respect to roles, e.g.,
    - maximum number of users assigned to a role
    - maximum number of roles a user can be assigned to
    - maximum number of roles that can be granted particular access rights
  - Prerequisite roles: Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role.
    - user can only be assigned a senior role if already assigned a junior role

$\texttt{RBAC}_3$: Combining $\texttt{RBAC}_1$ and $\texttt{RBAC}_2$

- $\texttt{RBAC}_3$ combines the role hierarchies of $\texttt{RBAC}_1$ with the constraints of $\texttt{RBAC}_2$, providing the most comprehensive and flexible RBAC model.
- It enables organizations to define complex permission structures while maintaining control over system configurations.

RBAC: Key Points

- Roles are collections of permissions, users, and possibly other roles (many-to-many)
- Role hierarchies simplify RBAC management and can be derived from organization structure
- Constraints prevent conflict of interest
- RBAC implementations simplify access control but may require role engineering
  - Role engineering for RBAC is the process of defining roles, permissions, constraints and role-hierarchies

- Case Study: RBAC for a Bank: All students: Optional Reading:
  - The Role-Based Access Control System of a European Bank: A Case Study and Discussion

Mandatory Security Policies

- MAC
- Multilevel Security (MLS) Models
- Bell-LaPadula Confidentiality Model
- Biba Integrity Model
- Chinese Wall Model

Why need MAC: Discretionary Access Control (DAC)

- An individual user can set an access control mechanism to allow or deny access to an object.
- Relies on the object owner to control access.
- Widely implemented in most operating systems, and we are quite familiar with it.
- Limitation of DAC
  - Global policy: DAC let users to decide the access control policies on their data, regardless of whether those policies are consistent with the global policies. Therefore, if there is a global policy, DAC has trouble to ensure consistency.
  - Information flow: information can be copied from one object to another, so access to a copy is possible even if the owner of the original does not provide access to the original copy. This has been a major concern for military.
  - Malicious software: DAC policies can be easily changed by owner, so a malicious program (e.g., a downloaded untrustworthy program) running by the owner can change DAC policies on behalf of the owner.
  - Flawed software: Similarly to the previous item, flawed software can be "instructed" by attackers to change its DAC policies.

Why need MAC: Mandatory Access Control (MAC)

- Definition
  - A system-wide policy decrees who is allowed to have access; individual user cannot alter that access.
  - Based on multilevel security (MLS)
- Relies on the system to control access.
- Examples
  - The law allows a court to access driving records without the owners' permission.
- Traditional MAC mechanisms have been tightly coupled to a few security models, often based on the Bell-LaPadula Model (BLP) or Biba Model.
- Recently, systems supporting flexible security models start to appear (e.g., SELinux, Trusted Solaris, TrustedBSD, etc.) offer MAC with more granular control capabilities.
  - These newer systems can incorporate additional factors beyond just clearance levels, such as user roles, specific data attributes, and even application contexts.
- "Strong" system security requires MAC
  - Normal users cannot be trusted

- *MAC is a security system designed to strictly control access to information based on its sensitivity and the user's security clearance*

Multilevel Security (MLS)

- MLS needed when subjects/objects at different levels access the same system.
- MLS is a specific type of access control designed for systems handling information with different classification levels (e.g., Top Secret, Confidential). It ensures that:
  - Users with appropriate clearances can access information at their level or lower.
  - Unauthorized access is prevented, even for users with access to the system itself.
- Military and government interest in MLS for many decades
  - Lots of research into MLS
  - Strengths and weaknesses of MLS well understood (almost entirely theoretical).
  - Many possible uses of MLS outside military

MLS Applications

- Classified government/military systems.
- Businesses
  - Restricting information to senior management only, all management, everyone in company, or general public
- Confidential medical info, databases, etc

Multilevel Security (MLS)

- Security Levels
  - People and information are classified into different levels of trust and sensitivity.
  - These levels represent the well-known security classifications:
    `TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED`
  - Clearance level indicates the level of trust given to a person with a security clearance, or a computer that processes classified information, or an area that has been physically secured for storing classified information. The level indicates the highest level of classified information to be stored or handled by the person, device, or location.
  - Classification level indicates the level of sensitivity associated with some information, like that in a document or a computer file. The level is supposed to indicate the degree of damage the country could suffer if the information is disclosed to an enemy.
  - Security level is a generic term for either a clearance level or a classification level

Classifications and Clearances

- Each subject and object has a security level
- Classifications apply to objects. Clearances apply to subjects.
- US Department of Defense (DoD) uses 4 levels:
  - TOP SECRET
  - SECRET
  - CONFIDENTIAL
  - UNCLASSIFIED

Subjects and Objects

- Let O be an object, S a subject.
  - O has a classification.
  - S has a clearance.
  - Security level denoted $L(O)$ and $L(S)$.
- For DoD levels, we have:

  `TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED`
- To obtain a `SECRET` clearance requires a routine background check
- A `TOP SECRET` clearance requires extensive background check

Example

Security Level	Subject	Object
Top Secret	Alice	Next Generation Designs
Secret	Bob	Marketing plans
Confidential	Carol	Last Quarter Financial Earnings
Unclassified	Dave	Telephone directory

MLS Security Models

- Security Policy Model
  - A security policy model is a succinct statement of the protection properties that a system, or generic type of system, must have.
- MLS models explain **what** needs to be done but **do not** tell you how to implement.
- Models are descriptive, not prescriptive.
  - That is, high level description, not an algorithm
- There are many MLS models. We will look at two simple models.
- Other models also more complex, more difficult to enforce, harder to verify, etc.

Bell-LaPadula Security Policy Model

- Proposed by David Bell and Len Lapadula in 1973, in response to U.S. Air Force concerns over the security of time-sharing mainframe systems.
- This model is the most widely recognized MLS model.
- Bell-LaPadula (BLP) is a security model designed to express essential requirements for MLS.
- i.e., is formal (mathematical) description of mandatory access control
- This model deals with confidentiality only
  - To prevent unauthorized reading.
- Recall that O is an object, S a subject.
  - O has a classification.
  - S has a clearance.
  - Security level denoted $L(O)$ and $L(S)$.

Bell-LaPadula

- Two properties: `No read up` and `no write down`
  - Simple Security Property: S can **read** O if and only if $L(O) \leq L(S)$.
  - *-Property (Star Property): S can **write** O if and only if $L(S) \leq L(O)$.

No Read Up

- ss-property: Simple Security Property
- Subject can only read an object of **less or equal** security level
  - $L(O) \leq L(S)$.

- i.e., to protect the confidentiality, you can not read information from the higher level

No write down

- *-Property
- A subject can only write[4] into an object of **greater or equal** security level.
  - $L(S) \leq L(O)$.
- Q: Why don't we allow subjects with higher clearance to write into files with lower clearance?

- The *-Property was Bell and LaPadula's critical innovation. It was driven by the fear that a user with "Secret" clearance might be "tricked" by attackers (e.g., through Trojan horse programs or software vulnerabilities) to copy down the information to a "Unclassified" area where the attackers can read.

[4] Here **write** is interpreted as "write-only" or "append". If **write** were to include both the ability to read and write then $L(S)=L(O)$ should be satisfied.

Bell-LaPadula

- Strong Star Property: an alternative to the *-Property, in which subjects may write to objects with only a matching security level.

DAC in MAC

- ds-property (discretionary security)
- A MAC system may also include a traditional discretionary access control check
- If *-property and simple security property checks pass, then also check the discretionary access rules
- Resource owner can deny access when the system allows it.
- Note that if the system denies access, user can't allow it!

BLP: The Bottom Line

- BLP is simple, probably too simple.
- BLP is one of the few security models that can be used to prove things about systems.
- BLP has inspired other security models.
  - Most other models try to be more realistic.
  - Other security models are more complex.
  - Models difficult to analyze, apply in practice

Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable to analyze, apply in practice
- Note relationship between integrity and trustworthiness
- Important point: integrity levels are NOT security levels

Biba's Model

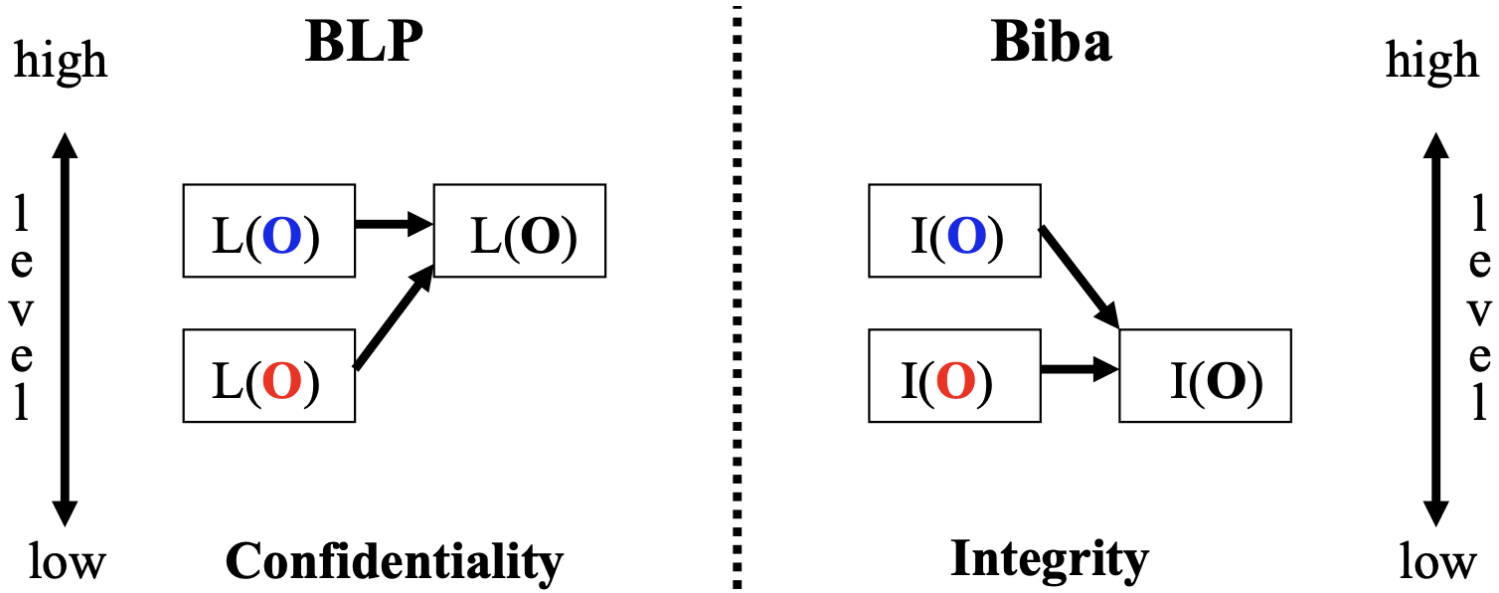
- Due to Ken Biba.
- Biba deals with integrity alone and ignores confidentiality entirely.
- Biba model covers integrity levels, which are analogous to sensitivity levels in Bell-LaPadula
- Rules about integrity levels prevent inappropriate modification of data and corruption caused by introducing unreliable information
- Prevents unauthorized users from making modifications (1st goal of integrity)
- Integrity model
  - Suppose you trust the integrity of 0 but not 0.
  - If object 0 includes 0 and 0, then you cannot trust the integrity of 0.
- Integrity level of 0 is **minimum** of the integrity of any objects in 0

Biba's Model

- Let $I(O)$ denote the integrity of object O and $I(S)$ denote the integrity of subject S
- Two properties:
  - Simple Integrity Property: S can write O if and only if $I(O) \leq I(S)$ (if S writes O, the integrity of $O \leq$ that of S)
    - A low integrity subject will not write or modify high integrity object (data). no write up
  - *-Property: S can read O if and only if $I(S) \leq I(O)$ (if S reads O, the integrity of $S \leq$ that of O)
    - The high integrity subject will not read low integrity object (data). no read down

- `Read Up, Write Down`: Subjects cannot read objects of lesser integrity, subjects cannot write to objects of higher integrity (`no read down, no write up`)

BLP vs Biba



- BLP (confidentiality): no read up, no write down.
- Biba (integrity): no read down, no write up.

The Chinese Wall Mode

- Focus is on conflicts of interest.
- Principle: Users should not access the confidential information of both a client organization and one or more of its competitors.
- Example: In a consultant company, a person who consult for BankOne should not have access to the data of BankTwo.
- How it works
  - Users have no “wall” initially.
  - Once any given file is accessed, files with competitor information become inaccessible.
- In this model, access control rules change with user behavior

Attribute-Based Access Control (ABAC)

- ABAC is a powerful and flexible authorization model that's gaining traction in modern security systems
- An ABAC model can define authorizations that express conditions on properties of both the resource and the subject.
- The true strength of ABAC lies in its ability to define fine-grained access control based on a rich set of attributes.
- This allows for highly customizable access rules that can precisely reflect an organization's security needs.
- For instance, consider a scenario where each resource has an "owner" attribute that identifies the user who created it.
  - Using ABAC, a single rule can efficiently grant ownership privileges to all creators across all resources within the system.
- Strength is its flexibility and expressive power

Attribute-Based Access Control (ABAC)

- Main obstacle to its adoption in real systems has been concern about the performance impact of evaluating predicates on both resource and user properties for each access.
- However, advancements in web services technologies are paving the way for efficient ABAC implementations, especially through the introduction of the eXtensible Access Control Markup Language (XAMCL)
  - XACML provides a standardized language for expressing ABAC policies, facilitating interoperability and potentially reducing the performance impact on access control decisions.

ABAC Model: Attributes

- With ABAC, an organization's access policies enforce access decisions based on the attributes of the subject, resource, action, and environment involved in an access event.
- Attributes can be about anything and anyone.
- Subject attributes
  - The subject is the user requesting access to a resource to perform an action.
  - Subject attributes are attributes that describe the user attempting the access
    - e.g. age, subject's identifier, name, organization, security clearance, department, role, job title, ...
- Object/Resource attributes
  - The resource is the asset or object (such as a file, application, server, tables, processes, programs, networks, domains) that the subject wants to access.
  - Resource attributes are all identifying characteristics, like a file's creation date, its owner, file name and type, and data sensitivity.
    - i.e., attributes that describe the object
    - e.g. the object type (medical record, bank account...), the department, the classification or sensitivity, the location...

ABAC Model: Attributes

- Action attributes
  - The action is what the user is trying to do with the resource.
  - Common action attributes include “read”, “write”, “edit”, “copy”, and “delete”.
  - In some cases, multiple attributes can describe an action.
    - e.g., requesting a transfer may have the characteristics “action type = transfer” and “amount = $1,000”
- Environment attributes
  - The environment is the broader context of each access request.
  - Attributes that deal with time and location of an access attempt, the subject’s device, communication protocol, and encryption strength.
  - These attributes have so far been largely ignored in most access control policies.

ABAC

- Distinguishable because it controls access to objects by evaluating rules against the attributes of entities, operations, and the environment relevant to a request
- Relies upon the evaluation of attributes of the subject, attributes of the object, and a formal relationship or access control rule defining the allowable operations for subject-object attribute combinations in a given environment
- Systems are capable of enforcing DAC, RBAC, and MAC concepts
- Allows an unlimited number of attributes to be combined to satisfy any access control rule

ABAC Policies

- A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions
- Typically written from the perspective of the object that needs protecting and the privileges available to subjects
- Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy
- Other terms commonly used instead of privileges are: rights, authorizations, and entitlements

ABAC Policies

- How does ABAC use attributes to express access control policies?[5]
- For example, let's say that the following policy is in place:

  `If the subject is in a communications job role, they should have read and edit access to media strategies for the business units they represent`

- Whenever an access request happens, the ABAC system analyzes attribute values for matches with established policies. As long as the above policy is in place, an access request with the following attributes should grant access:

```
Subject's  "job role" = "communications"
Subject's "business unit" = "marketing"
Action = "edit"
Resource "type" = "media strategy document"
Resource "business unit" = "marketing"
```

- ABAC allows admins to implement granular, policy-based access control, using different combinations of attributes to create conditions of access that are as specific or broad as the situation calls for.

[5] *Slide Credit:* https://www.okta.com/blog/2020/09/attribute-based-access-control-abac/

Summary

- What is the right access control model for my organization?[6]
- RBAC if:
  - You're in a small- to medium-sized enterprise
  - Your access control policies are broad
  - You have time to invest in a model that goes the distance
  - You have few external users, and your organization roles are clearly defined
- ABAC if:
  - You're in a large organization with many users
  - You want deep, specific access control capabilities
  - You have time to invest in a model that goes the distance
  - You need to ensure privacy and security compliance

[6] *Slide Credit:* https://www.okta.com/blog/2020/09/attribute-based-access-control-abac/

Key Points

- Access control to prevent unauthorized use of resources (objects) by subjects
- Subjects are processes on behalf of users and applications
- Classes of subjects: owner, group, world
- Objects: files, database records, disk blocks, memory segments, processes,...
- Access rights: read, write, execute, delete, create,...
- DAC: access rights may be granted to other subjects (common in operating systems and databases)
- RBAC: subjects take on role; access rights assigned to roles
- MAC: subjects/objects assigned to levels; subjects cannot modify assignment (e.g. military classification)

Security Issues

- Rely on correct assignment of capabilities/levels to subjects and objects by human administrator