

# Access Control: Implementation and Best Practices



By

Dr. Nawfal Fadhel

Contributors

Dr. Gary Wills

Dr. Faderica Paci

# Question

What is *Access*?

What is *Control*?

# What is Access Control

It is to give a trusted  
entity  
permission to access  
resource at  
some point in time.

It is to authenticate an  
entity,  
authorize it to access a  
resource and  
account for every action

# Question

Why is access control important?

# Importance of Access Control

## **Locally**

- It is used to differentiate inside users from outside users.
- help users to operate on their own personal data.

## **Network**

- It is used to give wide access to data and service.
- It holds all internet based services.
- Access control is achieved through means of (Access based defined) policies.

# Question

What are polices?

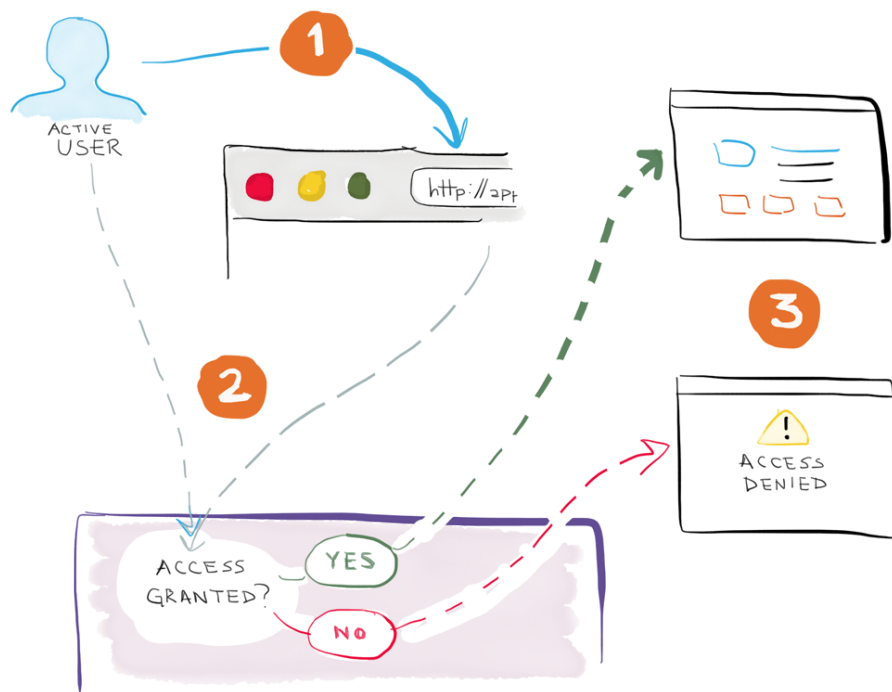
# Policy

Give X, permission to do Y within these  
constraints (such as time).

Examples:

# Web Access Control

- Web applications consist of elements that have different granularity levels and that are scattered through the entire application code.
- We need access control model and mechanism to deal with the distributed fine-grained access control code that protects the Web applications' elements.





# Web Access Control

- I. Database-oriented techniques where all access control is delegated to the database (eg. such as query rewriting to ensure people get only what they are entitled to see)
- II. Code-based approach that integrates the access control into the application code that is deployed on the server.

# Examples





## Database-oriented techniques

## Code-based approach






### Sharing settings

Link to share


<https://drive.google.com/drive/folders/1ehTwOFR5c0-WaiVIOxQzseg5M1Z9aqW?usp=sharing>

Share link via:    

Who has access

	Anyone who has the link can view	<a href="#">Change...</a>
	Bojan Siljanovski (you) bojan.s@gmail.com	is owner
	<div>help@coffeewithit.com</div>	 

Invite people:



Owner settings [Learn more](#)

☐ Prevent editors from changing access and adding new people

```
<?php
Class Acl {
    private $db;
    private $user_empty = false;

    //initialize the database object here
    function __construct() {
        $this->db = new db;
    }

    function check($permission,$userid,$group_id) {

        //we check the user permissions first
        If(!$this->user_permissions($permission,$userid)) {
            return false;
        }

        if(!$this->group_permissions($permission,$group_id) & $this->IsUserEm
            return false;
        }

        return true;
    }

    function user_permissions($permission,$userid) {
        $this->db->q("SELECT COUNT(*) AS count FROM user_permissions WHERE permis:
        $f = $this->db->f();

        If($f['count']>0) {
            $this->db->q("SELECT * FROM user_permissions WHERE permission_name='$per
            $f = $this->db->f();

            If($f['permission_type']==0) {
                return false;
            }

            return true;
        }
        $this->setUserEmpty('true');

        return true;
    }
}
```

# Importance of Access Control Design

- I. Access control models are used to:
  - I. Define a specific set of authorization rights.
  - II. Define a set of policies for a software system to enforce a set of rights to fulfil the security concerns.
  - III. Define a set of run-time system users that are used to assign the defined rights to the other users in the system.
  - IV. Protect for all multi-user systems, against violation of:
    - Confidentiality (eg. unauthorized disclosure)
    - Integrity (eg. improper modifications),
    - Availability (eg. Service disruption)

*Access control mechanism is the actual implementation of the access control model in the system.*

# Access Control Models

The main models of access control

- I. Discretionary (DAC)
- II. Mandatory (MAC).
- III. Role-based access control (RBAC) and its many flavours.

# Question

What dose the word Discretionary mean?

What dose the word Mandatory mean?

What dose the word Role based mean?

# What is the difference

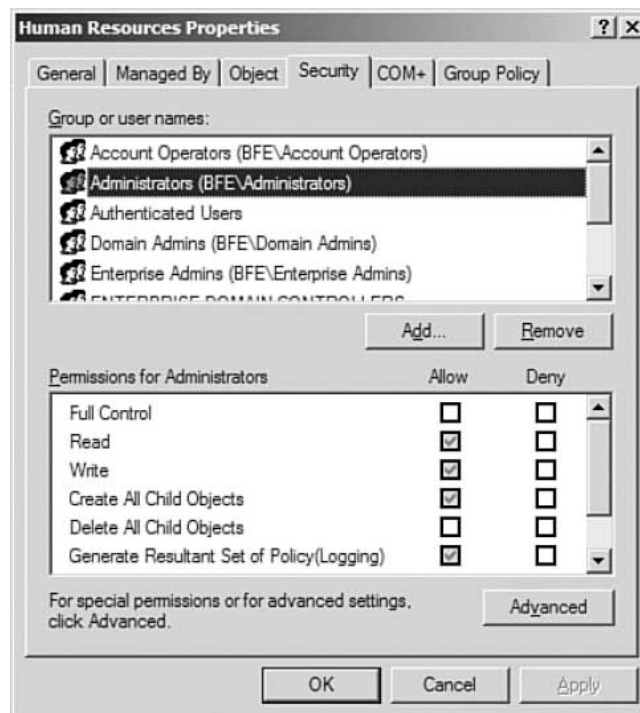
Discretionary is define by user.

Mandatory is define by a system.

Role Bases is define according to roles.

# Discretionary Access Control

- Probably the one you have come across the most.
  - Person want to perform an action on some data.
  - The access policy is consulted that check that that person has the access right to preform that operation.
  - Originated in operating systems
  - is often represented by the use of a matrix



The screenshot shows the 'File1: Permissions' dialog box with a table of permissions for Owner, Group, and Others.

	Read	Write	Execute
Owner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Others	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Octal: 755      Text: rwxr-xr-x

# Discretionary Access Control Issues

- As the access matrix represents the explicit access relation between each individual subject and object, it grows very large very quickly,
  - but remains sparse as most subjects remain unrelated to
  - This also leads to a complex authorization management.
  - The main drawback is that they are unable to enforce any sort of control of the flow of information from the process that is operating on behalf of the user .
  - This allows "Trojan Horse" processes to leak information



# Discretionary Access Control

- In the DAC context it is assumed that every object has an owner that controls the permissions to access the object.
- Owners have the ability to make policy decisions and/or assign security attributes.
- A straightforward example is the Unix file mode
  - Chmod 777 or Chmod 775

	Owner	Group	Everyone
0 – no permission	0	0	0
1 – execute	1	1	1
2 – write	2	2	2
3 – write and execute	3	3	3
4 – read	4	4	4
5 – read and execute	5	5	5
6 – read and write	6	6	6
7 – read, write, and execute	7	7	7

# Discretionary Access Control

- Chmod John Video\_folder 777
- Example
  - John wants her friend Alice to have view access on her photo album folder”Spanish Holidays”
  - John wants to share only with his wife his note.txt  
”Happy like the first day! Happy Anniversary my love!”
  - John wants to share with all his friends a funny video of her cat

	Owner	Group	Everyone
0 – no permission	0	0	0
1 – execute	1	1	1
2 – write	2	2	2
3 – write and execute	3	3	3
4 – read	4	4	4
5 – read and execute	5	5	5
6 – read and write	6	6	6
7 – read, write, and execute	7	7	7

# DAC Limitations

- Managing a policy is a complex task in a large system
  - Set of subjects or objects is large
  - Set of subjects or objects change frequently
- Capability Lists
  - Difficult to get an overview of permissions granted on a given object
- Access Control Lists
  - Difficult to get an overview of permissions granted to a given user

# Mandatory Access Control

- The most widely used MAC policy is the multi-level security policy model.
  - A process on behalf of the user, tries to access the object (note the user cannot access directly)
  - Allow MAC to also control indirect access from executed processes.
- There are four levels to multilevel security
  - I. Top Secret
  - II. Secret
  - III. Confidential
  - IV. Unclassified

# Mandatory Access Control

- MAC is a security strategy that restricts the ability individual resource owners have to grant or deny access to resource objects in a file system
  - Owners (users) cannot change these permissions.
- The policy are enforced by the operating systems or security kernel.

# Mandatory Access Control

- A security policy model is a succinct statement of the protection properties which a system, or generic type of system, must have.
  - Example: IoT, Blockchain, Cloudbased system
- Security policy is a document that expresses clearly and concisely what the protection mechanisms are to achieve.
- Typically it says: which user may access which data.
  - Example: DoomCorp Security policy
    1. Data shall be available only to those with ‘need to know’
    2. Anyone who breaches this policy shall be shot!!

# Mandatory Access Control

- When an entity attempts to access a specific resource, the OS or security kernel will check the entity's credentials to determine whether access will be granted.
- MAC requires careful planning and continuous monitoring to keep all resource objects' and users' *classifications* up to date.

# Summary

- We use access control to protect an applications' elements on a distributed systems, like the Web.
- Discretionary Access Control owners have the ability to make policy decisions and/or assign security attributes.
  - the most easiest to implement



# Summary

- Mandatory Access Control are enforced by the operating systems or security kernel
- MAC requires careful planning and continuous monitoring to keep all resource objects' and users' classifications up to date

# Role Based Access control

- In this class of access control models (grouping privileges)
  - privileges are collected based on common aspects,
  - then authorizations are assigned to these collections
- The advantage of using grouping privilege based access control models such as RBAC, is that
  - Compared to DAC or MAC models,
  - This class the access control **factors out similarities**, and so handles changes in the system better, which leads to easier authorization management

# Role Based Access control

- Baldwin in 1990 introduced the name protection domain (NPD)
  - the set of all privileges that need to exist to do each task in the system (grouping of privileges)
  - Groups were assigned to each user based on the required policy
- RBAC is the most widely used group privileges class since its introduction by Kuhn and Gerraiolo in 1992, it is used in
  - Operating systems.
  - Web security and related technologies.
  - Distributed systems, databases.
  - Embedded systems.
- RBAC was standardized by the National Institute of Standards and Technology (NIST) in 2001

# Role Based Access control

- RBAC concept uses the notion of "role" as the central authorisation mechanism
  - a role is an abstract representation of a group of subjects that are allowed to perform the same operations on the same objects
  - a (formal) concept using concept lattices
  - the objects (i.e., accessible shared data) in the system are assigned to an authorised role
  - the subjects (i.e., users) need to identify themselves to acquire these roles to access and operate on the objects

# Role Based Access control:

## Example

- University system can use lecturers and student roles to represent the authorization rights of the lecturers and students
- The role lecturer could be associated with two permitted operations (reading and writing on the mark object)
- While the subjects associated with the student role can just see their own mark.

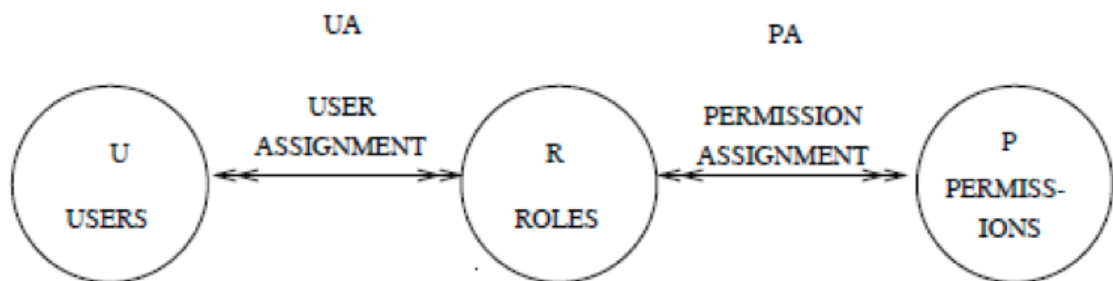
# Role Based Access control

- RBAC uses the concept of "role", as a result it fits naturally in to an organisational structure
  - leading to an easier design and development of the access control mechanism
- As roles are changed less frequent then the people in these roles
  - the role as a central authorization element decreases the maintenance cost and administrative tasks

# Flat RBAC

- Flat RBAC, is the base level of RBAC, defines the minimal requirements for an RBAC approach as an access control part of the authorization mechanisms of a system.
  - The central element is the role, and users and permissions (i.e., relations between the data and operations) have one-to-many or many-to-many relations to roles
  - In addition to this, the user has to be able to review (but not necessary change) his/her assigned roles and to use a number of permissions from multiple roles simultaneously
  - A very simple example of the flat RBAC would be a system that has lecturer and student as roles, and reading or writing of student marks as permissions.

# Flat RBAC Role Based Access control

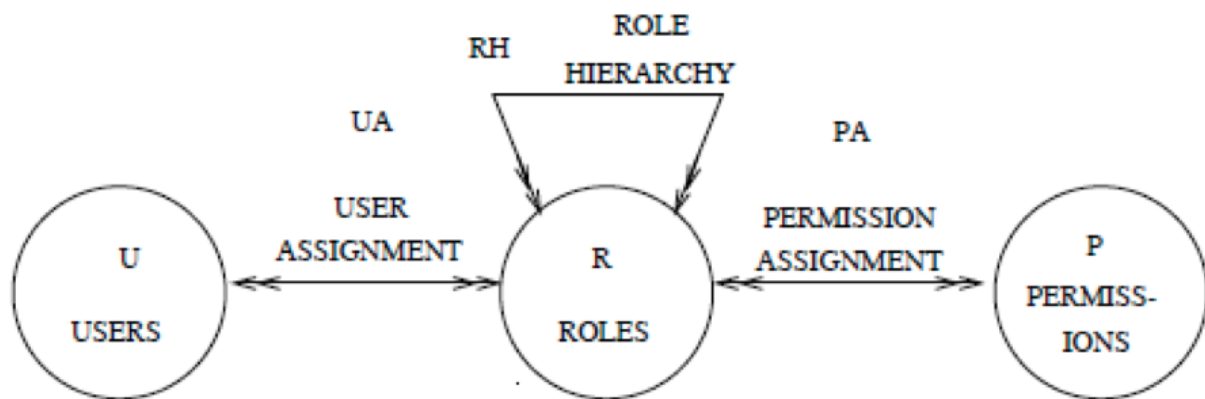




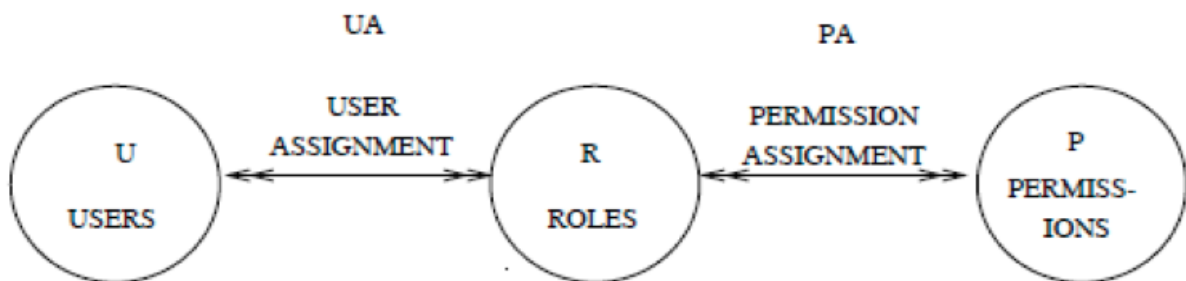
# Hierarchical RBAC

- hierarchical RBAC, is built on top of flat RBAC
- Hierarchical RBAC adds the notion of role hierarchy.
  - The role hierarchy component could be of type arbitrary or limited.
  - By using the limited hierarchy means, the model can assign a set of authorization rights to a part of the roles' hierarchy
- For example, the supervisor inherits the rights given to the lecturer role but can have, in addition, further rights, e.g., the authorization to access for reading each student's medical history who is under her supervision.

# Hierarchical RBAC



Flat



# Constrained RBAC

- Constrained RBAC introduces an access control element called separation of duties (SOD) on role hierarchies and user-role assignments
  - main design principles of access control, improper data modification is not desirable and must thus be prevented
  - the administrator should not be allowed to assigned a number of conflicting components (e.g., roles).
  - conflict components means the components that should not be used together.

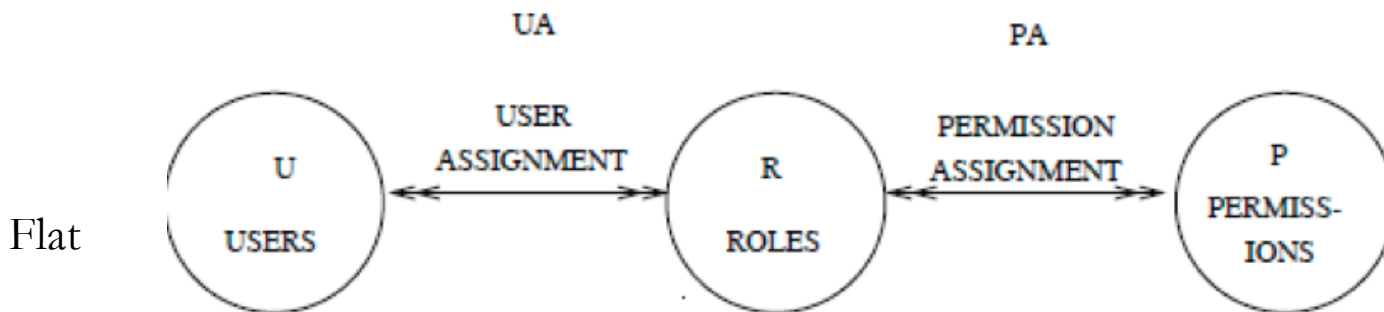
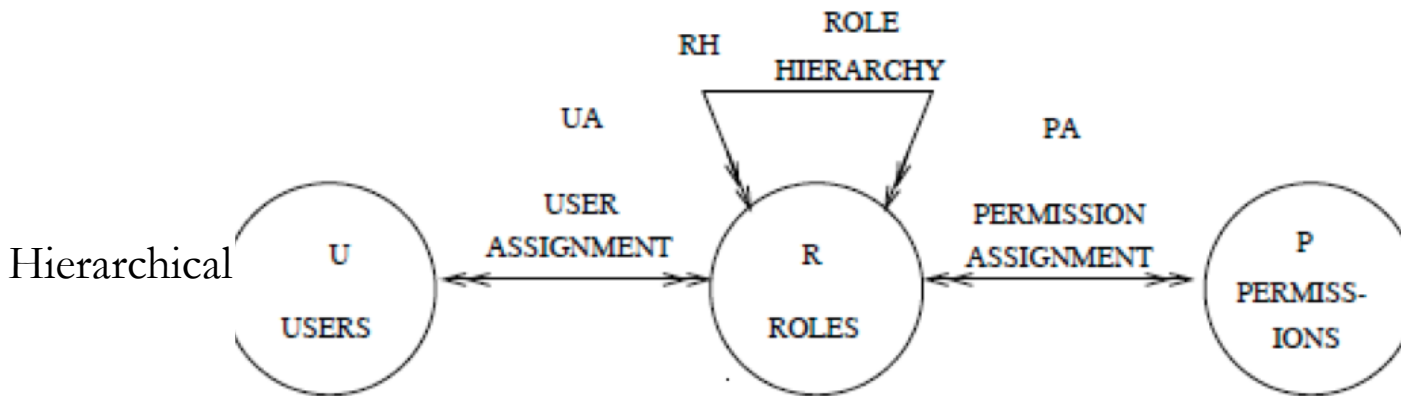
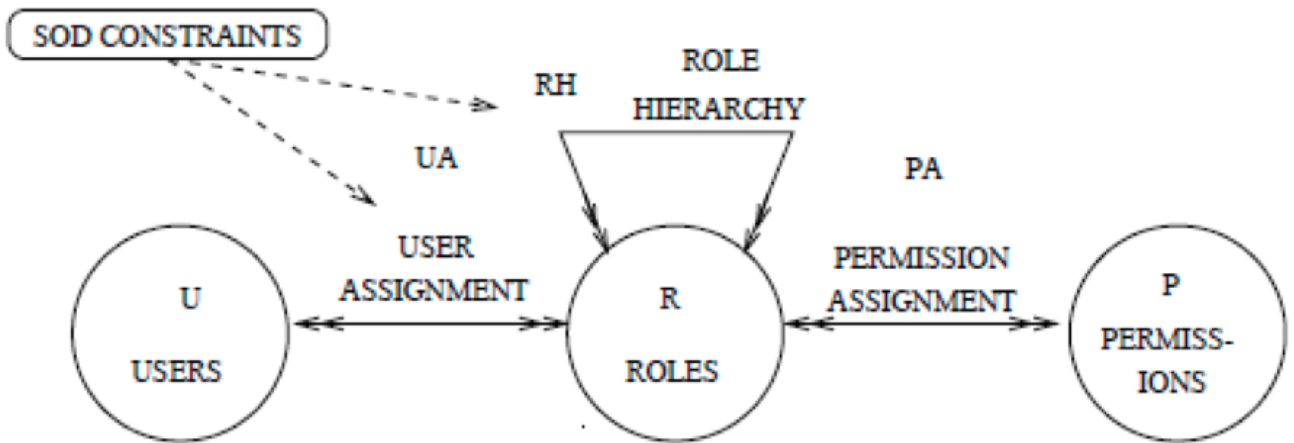
# Constrained RBAC

- As well as just improperly assigning a role, giving someone too many authorisation rights can lead to unintended access to material they should not have access to.
  - SOD can be used to divide the rights among the subjects based on the constraints and/or conflicts that might exist among the roles [
  - The SOD concept itself originated in the data-integrity domain, to stop improper behaviour of the subjects

# Constrained RBAC

- Static SOD: affects the allowed role hierarchies and user assignments.
  - Static SOD enforces on the user-role assignments that if two roles always create a security conflict, then these roles cannot be assigned to the same subject at any given time.
- For example,
  - consider the lecturer and student roles.
  - Normally, a student submits their work and a lecturer marks it.
  - In this case if a subject (i.e., user) has both authorization rights, they can submit and edit their own mark, which is prohibited.
  - This can be modelled by a static SOD constraints.

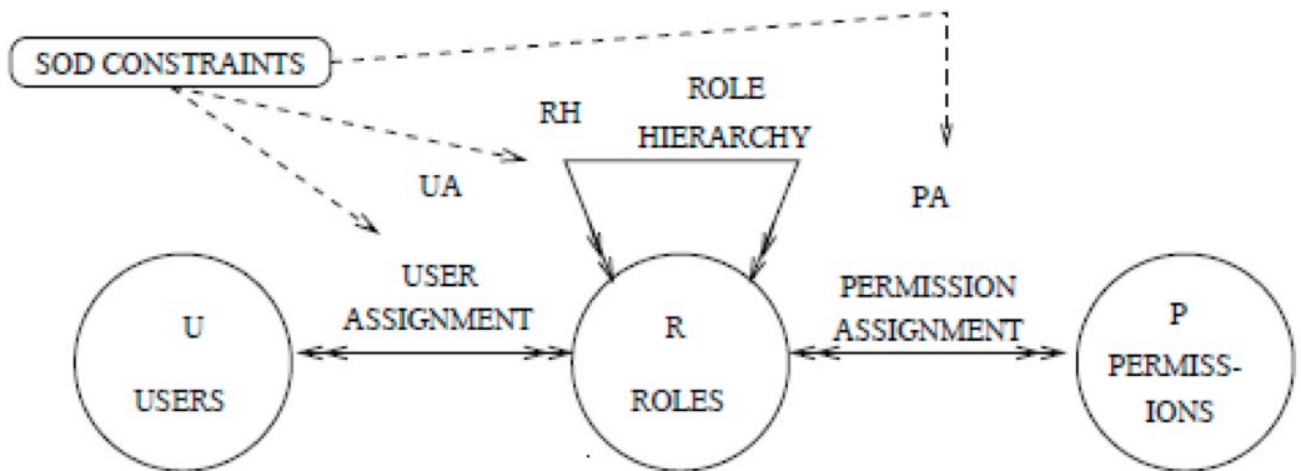
# Constrained RBAC-Static SOD



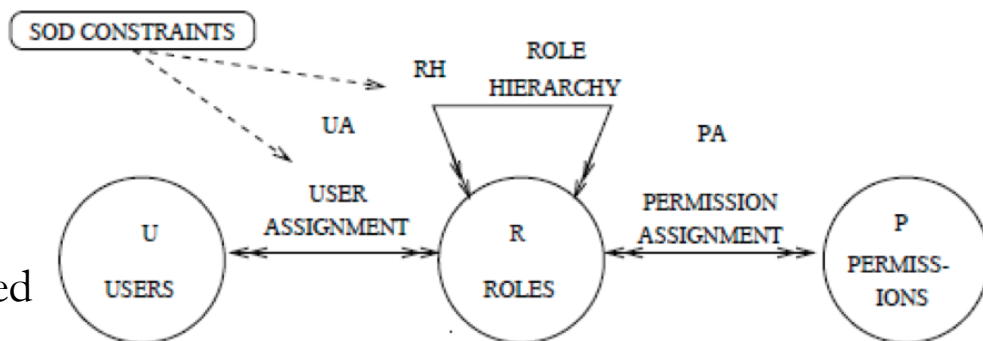
# Constrained RBAC

- Dynamic SOD: affects each user session.
  - After user authentication, the system creates a session for the user.
  - Dynamic SOD enforces the separation of authorization rights within each user session.
- For example, consider the policy that states a subject cannot be the examiner of a student they are supervising.
  - In this case, there must be a dynamic SOD between the roles supervisor and examiner, but only if the student they are supervising needs to be examined.
  - In all other circumstances, there is no conflict between the supervisor and examiner roles.

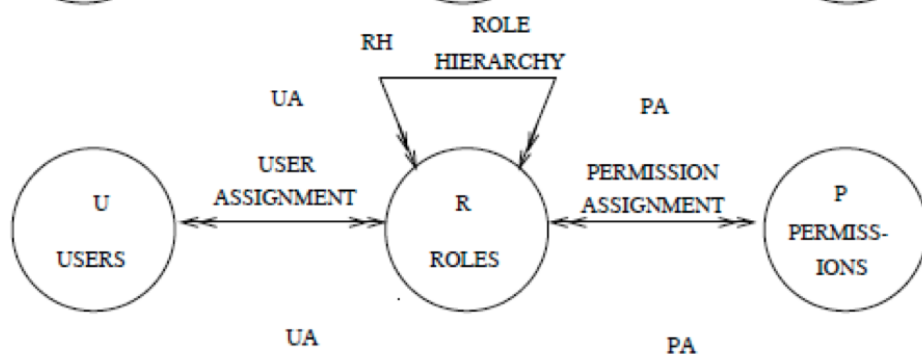
# Dynamic SOD



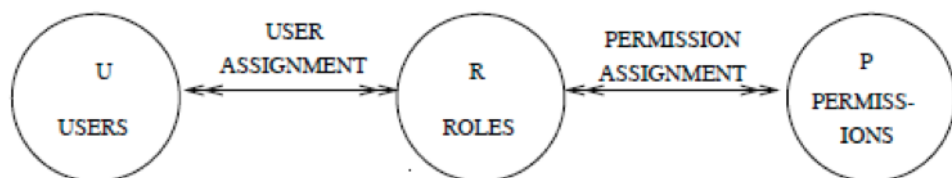
Constrained



Hierarchical



Flat





# Constrained RBAC

- The difference of the separation of duties can make in terms of developing and using such an RBAC system, is to introduce static and dynamic checks on the conflict roles.
  - These checks make sure that the conflicting roles do not get assigned to the users.
  - In case of static SOD, the administrator cannot assign conflicted roles to the same subject.
  - In the case of dynamic SOD the administrator can assign both roles to a user but the user cannot activate both conflicted roles in one session.

# Symmetric RBAC

- In this level the notion of separation of duty constraints extended to constrain permission assignments as well.
- Therefore, in this level, conflicting permissions cannot be assigned to the same role

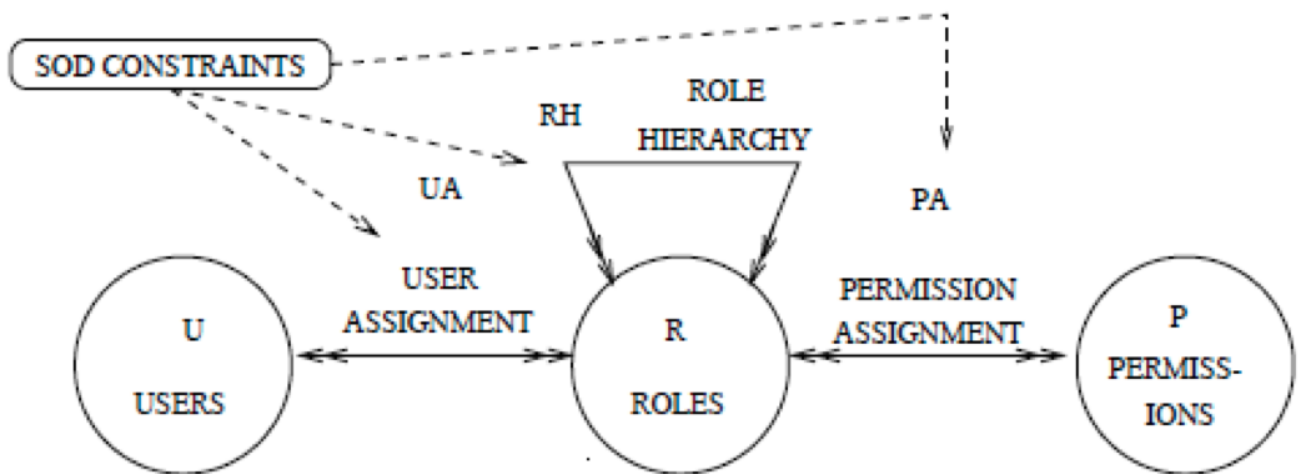
# Symmetric RBAC

- Static SOD:
  - constraints that always create conflicts between permission-role assignments need to be separated such that under no circumstances conflicted permissions would be assigned to the same role.
  - For example, creating and deleting of a bank account should not be possible for any user at the same time.
  - Therefore the creating and deleting permissions cannot be assigned to the same role.
  - Therefore, in Symmetric RBAC, the negative permissions are specified, separated and assigned to different roles.

# Symmetric RBAC

- Dynamic SOD:
- Here the user session is taken into account again, such that if two permissions are conflicting only when they are used together, then these two permissions could be assigned to a role but they cannot be acquired together during any user's session.
- For an example
  - consider a policy that states a lecturer cannot be the second marker of an exam if they marked the papers for the first time.
  - Therefore, clearly in this case during the user session, if the lecturer attempts to mark the papers then they cannot use the same role to give the mark for the second time.
  - The interesting point here is that the conflict is not on two different permissions but on the same permission that relates to the role Lecturer . Therefore, if a user uses their lecturer role for a course to mark the papers, they cannot use this role again in the same session to mark the papers for the second time.

# Symmetric RBAC



# Summary

Level	Type
Flat	<p>RBAC</p> <ol style="list-style-type: none"><li>1. Must allow users to acquire permissions through roles.</li><li>2. Must support many-to-many user-role assignment.</li><li>3. Must support many-to-many permission-role assignment.</li><li>4. Must support user-role assignment review.</li><li>5. Use permissions of multiple roles simultaneously.</li></ol>
Hierarchical	<p>Flat RBAC +</p> <ol style="list-style-type: none"><li>1. Must support role hierarchy (partial order).</li><li>2. Level 2S Requires support for arbitrary hierarchies.</li><li>3. Level 2D Requires support for limited hierarchies.</li></ol>
Constrained	<p>Hierarchical RBAC +</p> <ol style="list-style-type: none"><li>1. Must enforce separation of duties (SOD).</li><li>2. Level 3S Requires support for arbitrary hierarchies.</li><li>3. Level 3D Requires support for limited hierarchies.</li></ol>
Symmetric	<p>Constrained RBAC +</p> <ol style="list-style-type: none"><li>1. Must support permission-role review with performance effectively comparable to user-role review.</li><li>2. Level 4S Requires support for arbitrary hierarchies.</li><li>3. Level 4D Denotes support for limited hierarchies</li></ol>

# Questions

- Why is access control required.
- What advantages does DAC have over RBAC and visa-versa.
- What type of RBAC is required for (and Why):
  - A hospital
  - A military establishment
  - A research facility
  - A factory

# Other Sources

- <http://www.cs.cornell.edu/fbs/publications/chptr.DAC.pdf>
- <http://www.cs.cornell.edu/fbs/publications/chptr.MAC.pdf>