Chapter 2

Preliminaries

2.1 Access Control Features

While designing and implementing an access control system, several factors need to be considered. Here we list some of the features that play a vital role in providing effective security.

- Flexibility: Generally, access control requirements keep changing. Therefore, it is essential that an access control model is flexible enough to accommodate such changes easily.
- Manageability: Administration of the model should not be an overhead, i.e., adding, updating, and deleting entities in the system should be easy to do and should not create an overhead.
- Dynamic: The model should be able to take access decisions based on real-time access

parameters. This is becoming increasingly important recently as systems require access decisions based on dynamic parameters such as the location of the user, time, and day of access.

- Scalability: A model should be able to accommodate an increase in users, objects, permissions etc., easily. An increase in the system entities should not cause much overhead on configuration, management, and performance.
- Auditability: It is important to be able to determine the sets of permissions granted to a particular user or sets of users that have been granted a particular permission. This is essential to evaluate the system security, risk assessment, and detecting the source of the attack in case of a breach.
- Granularity: Granularity of the access control policy represents how accurately we can control the access to the resource. Fine-grained access control helps in specifying precise access rules and is essential to achieve *principle of least privilege*.
- Reliability: It should be able to ensure that the authorization decisions are as per the security requirements.
- Performance: While providing strong security is essential, achieving this with minimal performance overhead is also important.

2.2 Access Control Models

In this section, we provide brief descriptions of four well-known access control models, DAC, MAC, RBAC, and ABAC.

2.2.1 Discretionary Access Control

In discretionary access control (DAC), the access permissions are decided based on the user identity and the permissions specified by the object's owner. A DAC system can be described using the access matrix model Lampson (1974). The model was formalized by Harrison, Ruzo, and Ullmann (HRU) Harrison et al. (1976).

	File1	File2	Program1
Amar	own read write	read	
Bina		write	execute

Table 2.1: Access Matrix

In the access matrix model, the system security state is represented by a matrix M, where rows correspond to users and columns correspond to objects, and cell $M[u,o]$ specifies the privileges of user u on object o. Table 2.1 shows a simple access matrix with two users and three objects. As shown in the table, depending on the type of the object, various privileges are given to users. Apart from privileges like read, write, and execute, users can also get the *ownership* privilege of an object o that allows them to change entries in the column corresponding to o.

Although access matrix is an easy and intuitive way to represent authorization, in practice, the permissions are not stored in this format. Generally, in large systems, the access matrices are large and sparse (i.e., most of the cells would be empty) and would require more memory. Two common ways to address this are to store the permissions either using Access Control List (ACL) or Capability List (CL). ACL represents the objects' perspective of the matrix and stores the privileges using individual object columns describing the set of users who have access to a particular object. CL represents users' view of the matrix and stores the privileges using individual rows describing the set of privileges of a user. Each of these methods has its own benefits. In ACL, it is easy to find the set of users that can access a particular object, whereas, in CL, it is easy to find all the privileges available to a user Jaeger (2008). Below are some of the major advantages and shortcomings of DAC:

Advantages of DAC:

- DAC provides a simple, user-centric approach to specify permissions. Due to this simplicity, it is widely used in commercial systems, including many major operating systems.
- DAC mechanisms are also flexible. They support easy specification and modifications of permissions.

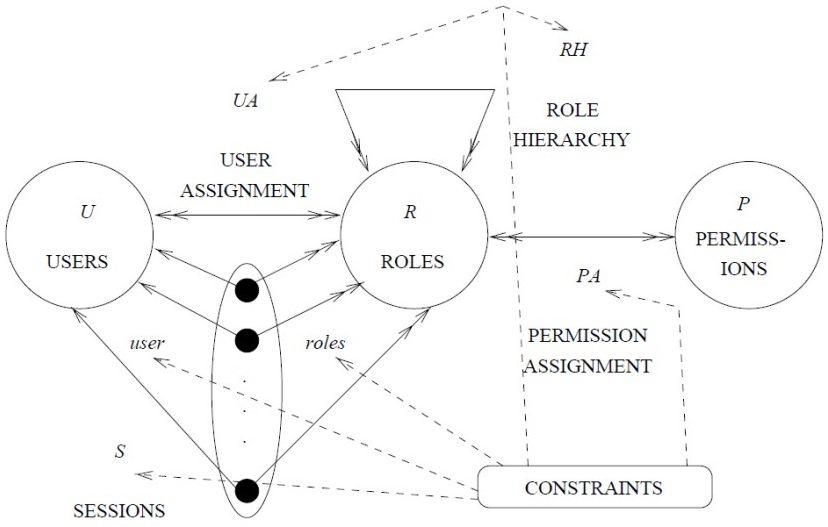

Figure 2.1: RBAC Model Sandhu et al. (1996)

Shortcomings of DAC:

- The privileges are decided by the objects' owners rather than through a system-wide policy.
- As the number of users increase, assigning and managing permissions becomes challenging.
- Due to its inherent limitations, DAC is more vulnerable to Trojan horse attacks.

2.2.2 Role-Based Access Control

RBAC was first formalized in 1992 by the National Institute of Standards and (NIST) David and Richard (1992). Ever since several variations have been proposed. It was standardized in 2004 American National Standard for Information Technology (2004) and was later revised in 2012 INCITS (2012).

Unlike DAC, where permissions are directly assigned to users, in RBAC, permissions are assigned to roles, and then the roles are assigned to users. This introduction of a level of indirection greatly simplifies the policy specification as well as its maintenance. It provides several benefits such as scalability, ease of administration, and auditability Sandhu et al. (2000);

Kuhn et al. (2010). So naturally, RBAC has been one of the most widely implemented access control models.

Figure 2.1 shows the components in an RBAC model. Here users (U) are mapped to roles (R) using a function called User Assignment (UA). It is a many-to-many function, i.e., each user u can be mapped to multiple roles R_u and each role r can be assigned to multiple users. Similarly, permissions are assigned to roles through a many-to-many function called Permission Assignment (PA). A session can be associated with exactly one user, and it remains unchanged throughout the session's lifetime. On session creation, a subset of the user's roles can be activated. Valid permissions in a session are the union of permissions available to all the active roles in the session.

Roles can be structured hierarchically where senior roles inherit the permissions assigned to their junior roles. As a result, effective permissions of a role include the direct permissions assigned to it through PA and the inherited permissions. Role hierarchy provides an intuitive way to structure the RBAC roles to represent the organizational/authorization hierarchy.

RBAC also supports specifying constraints on various components of RBAC configuration. Generally, constraints are specified with respect to user assignment, permission assignment, and session creation. Constraints help in achieving *Separation of Duty*. With the help of constraints on user assignment, we can achieve static separation of duty, and using constraints on permission assignment, we can achieve dynamic separation of duty. Below are some of the major advantages and shortcomings of RBAC:

Advantages of RBAC:

- With role and role hierarchies, RBAC can easily capture an organization's structure.
- RBAC policies are easy to manage and scale Sandhu et al. (2000) Kuhn et al. (2010).
- Supports separation-of-duty
- Since the permissions are assigned to users through role mapping, it is easy to find details such as the set of permissions available to users at any point of time and the set of users with a particular permission Kuhn et al. (2010). This helps in risk assessment as well as incident analysis.

Shortcomings of RBAC:

- Creating RBAC policies through role-engineering is a complex and resource-consuming task.
- Since the permissions are assigned through roles, the policies are coarse-grained. Attempt for finer granularity usually leads to role explosion Hu et al. (2013).
- RBAC doesn't support authorization based on parameters such as time, day, location etc., making it challenging to use it in dynamic environments Hu et al. (2013).

2.2.3 Mandatory Access Control

Mandatory Access Control is another traditional access control model where subjects and objects are labeled, and access rules are specified in terms of these labels. The access decisions are taken based on policies (set of rules) specified by the organization, and permissions cannot be changed at the user's discretion. Instead, all the changes have to adhere to the policy.

Multi-level models are the most common MAC models. These models were originally developed for use in the military that required strict information flow control. The classic MAC models BLP Bell and LaPadula (1973) and Biba Biba (1977) are multi-level security (MLS) models where the labels are ordered and form a lattice. BLP provides confidentiality through its *no-read-up* and *no-write-down* policy. Biba is a dual of BLP and provides integrity. Denning's model Denning (1976) is another classic lattice-based MAC model that provides information flow control. Although these models provide essential security features, they are not widely used in mainstream systems. This is mainly due to the fact that these models are not flexible enough to be used in mainstream systems without hindering their usability. Over the years, several variations of these models have been proposed to make them more flexible and applicable for mainstream systems.

Apart from multi-level models, other well-known MAC models include the Chinese-Wall model Brewer and Nash (1989), Clark-Wilson Model Clark and Wilson (1987), and Domain and Type Enforcement Badger et al. (1995). In DTE, processes are grouped into *domains*, and files into *types*, and access rules are specified in terms of these types and domains.

Since there are many MAC models, each with different characteristics, their advantages

and shortcomings vary. With non-discretionary access control, they generally provide better confidentiality and/or integrity. However, they generally tend to be less flexible and require considerable effort in the initial configuration.

2.2.4 Attribute-Based Access Control

ABAC is an emerging access control model where access rules are specified in terms of attributes. Attributes are nothing but key-value pairs that represent the characteristics or status of the entities or their environment. The access rules can be specified using logical formulas, or through enumeration Biswas et al. (2016). With the use of attributes, ABAC enables expressing precise access rules specifying which subject can access which object under what conditions, when, and where.

As ABAC is a relatively new model, there has been less work on standardization with respect to policy specification and enforcement. The National Institute of Standards and Technology has published a guide Hu et al. (2013) to ABAC definition and considerations. It provides the basic concepts of ABAC and considerations for the adoption of ABAC in an enterprise. However, it does not provide a formal model. Using attribute-based rules gives more expressive power to ABAC. Jin et al. Jin et al. (2012a) presented a unified ABAC model called $ABAC_\alpha$. The authors provide a basic formal model that can cover traditional models, DAC, MAC, and RBAC.

We list some of the major benefits and limitations of ABAC below:

Advantages of ABAC:

- With simple and intuitive access rules, ABAC policies are flexible and easy to modify whenever there is a change in security requirements Hu et al. (2013).
- ABAC supports authorization rules based on dynamic parameters like time, location, etc.
- With the support for rules based on attributes, ABAC provides fine-grained access control Hu et al. (2013).