Access control is very crucial for sensitive content since it ensures the selective exposure of information to different classes or groups of users. Access Control is an important component of Cloud Computing; specially, User access control management; however, Access Control in Cloud environment is different from traditional access environment and using general access control model can't cover all entities within Cloud Computing. Cloud environment includes different entities such as data owner, end user, and service provider.

Authorization is the process where requests to access a particular resource should be granted or denied. It should be noted that authorization is not equivalent to authentication - as these terms and their definitions are frequently confused. Authentication is providing and validating identity. Authorization includes the execution rules that determines what functionality and data the user (or Principal) may access, ensuring the proper allocation of access rights after authentication is successful.

Web applications need access controls to allow users (with varying privileges) to use the application. They also need administrators to manage the applications access control rules and the granting of permissions or entitlements to users and other entities. Various access control design methodologies are available. To choose the most appropriate one, a risk assessment needs to be performed to identify threats and vulnerabilities specific to your application, so that the proper access control methodology is appropriate for your application.

**Role Based Access control model**

In Role-Based Access Control (RBAC), access decisions are based on an individual's roles and responsibilities within the organization or user base.

The process of defining roles is usually based on analyzing the fundamental goals and structure of an organization and is usually linked to the security policy. For instance, in a medical organization, the different roles of users may include those such as doctor, nurse, attendant, nurse, patients, etc. Obviously, these members require different levels of access in order to perform their functions, but also the types of web transactions and their allowed context vary greatly depending on the security policy and any relevant regulations (HIPAA, Gramm-Leach-Bliley, etc.).

An RBAC access control framework should provide web application security administrators with the ability to determine who can perform what actions, when, from where, in what order, and in some cases under what relational circumstances.

The advantages of using this methodology are:

* Roles are assigned based on organizational structure with emphasis on the organizational security policy
* Easy to use
* Easy to administer
* Built into most frameworks
* Aligns with security principles like segregation of duties and least privileges

Problems that can be encountered while using this methodology:

* Documentation of the roles and accesses has to be maintained stringently.
* Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
* There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for. Or a user might be included in two roles if proper access reviews and subsequent revocation is not performed.
* Does not support data based access control

The areas of caution while using RBAC are:

* Roles must be only be transferred or delegated using strict sign-offs and procedures.
* When a user changes his role to another one, the administrator must make sure that the earlier access is revoked such that at any given point of time, a user is assigned to only those roles on a need to know basis.
* Assurance for RBAC must be carried out using strict access control reviews.

**Discretionary Access Control (DAC)**

Discretionary Access Control (DAC) is a means of restricting access to information based on the identity of users and/or membership in certain groups. Access decisions are typically based on the authorizations granted to a user based on the credentials he presented at the time of authentication (user name, password, hardware/software token, etc.). In most typical DAC models, the owner of information or any resource is able to change its permissions at his discretion (thus the name).

A DAC framework can provide web application security administrators with the ability to implement fine grained access control. This model can be a basis for data based access control implementation

The advantages of using this model are:

* Easy to use
* Easy to administer
* Aligns to the principle of least privileges.
* Object owner has total control over access granted

Problems that can be encountered while using this methodology:

* Documentation of the roles and accesses has to be maintained stringently.
* Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
* There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for.

The areas of caution while using DAC are:

* While granting trusts
* Assurance for DAC must be carried out using strict access control reviews.

**Mandatory Access Control (MAC)**

Mandatory Access Control (MAC) ensures that the enforcement of organizational security policy does not rely on voluntary web application user compliance. MAC secures information by assigning sensitivity labels on information and comparing this to the level of sensitivity a user is operating at. MAC is usually appropriate for extremely secure systems including multilevel secure military applications or mission critical data applications.

The advantages of using this methodology are:

* Access to an object is based on the sensitivity of the object
* Access based on need to know is strictly adhered to and scope creep has minimal possibility
* Only an administrator can grant access

Problems that can be encountered while using this methodology:

* Difficult and expensive to implement
* Not agile

The areas of caution while using MAC are:

* Classification and sensitivity assignment at an appropriate and pragmatic level
* Assurance for MAC must be carried out to ensure that the classification of the objects is at the appropriate level.

**Permission Based Access Control**

The key concept in Permission Based Access Control is the abstraction of application actions into a set of permissions. A permission may be represented simply as a string based name, for example "READ". Access decisions are made by checking if the current user has the permission associated with the requested application action.

The has relationship between the user and permission may be satisfied by creating a direct relationship between the user and permission (called a grant), or an indirect one. In the indirect model the permission grant is to an intermediate entity such as user group. A user is considered a member of a user group if and only if the user inherits permissions from the user group. The indirect model makes it easier to manage the permissions for a large number of users, since changing the permissions assigned to the user group affects all members of the user group.

In some Permission Based Access Control systems that provide fine-grained domain object level access control, permissions may be grouped into classes. In this model it is assumed that each domain object in the system can be associated with a class which determines the permissions applicable to the respective domain object. In such a system a "DOCUMENT" class may be defined with the permissions "READ", "WRITE" and DELETE"; a "SERVER" class may be defined with the permissions "START", "STOP", and "REBOOT".

**Attribute-based access control**

(ABAC) defines an [access control](https://en.wikipedia.org/wiki/Access_control) paradigm whereby access rights are granted to users through the use of [policies](https://en.wikipedia.org/wiki/Policy) which combine attributes together. The policies can use any type of [attributes](https://en.wikipedia.org/wiki/Attribute_(computing)) (user attributes, resource attributes, object, environment attributes etc.). This model supports [Boolean logic](https://en.wikipedia.org/wiki/Boolean_Logic), in which rules contain "IF, THEN" statements about who is making the request, the resource, and the action. For example: IF the requestor is a manager, THEN allow [read/write access](https://en.wikipedia.org/wiki/File_system_permissions) to sensitive data.[[1]](https://en.wikipedia.org/wiki/Attribute-Based_Access_Control#cite_note-1)

Unlike [Role-Based Access Control (RBAC)](https://en.wikipedia.org/wiki/Role-based_access_control), which employs pre-defined roles that carry a specific set of [privileges](https://en.wikipedia.org/wiki/Privilege_(computing)) associated with them and to which subjects are assigned, the key difference with ABAC is the concept of policies that express a complex Boolean rule set that can evaluate many different attributes.[[2]](https://en.wikipedia.org/wiki/Attribute-Based_Access_Control#cite_note-2)Attribute values can be set-valued or atomic-valued. Set-valued attributes contain more than one atomic value. Examples are *role* and *project*. Atomic-valued attributes contain only one atomic value. Examples are [clearance](https://en.wikipedia.org/wiki/Classified_information#Clearance) and [sensitivity](https://en.wikipedia.org/wiki/Information_sensitivity). Attributes can be compared to static [values](https://en.wikipedia.org/wiki/Value_(computer_science)) or to one another, thus enabling relation-based access control.

Although the concept itself existed for many years, ABAC is considered[[3]](https://en.wikipedia.org/wiki/Attribute-Based_Access_Control#cite_note-3) "next generation" authorization model because it provides dynamic, context-aware and risk-intelligent access control to resources allowing access control policies that include specific attributes from many different information systems to be defined to resolve an authorization and achieve an efficient regulatory compliance, allowing enterprises flexibility in their implementations based on their existing infrastructures.

Attribute-Based Access Control is sometimes referred to as Policy Based Access Control (PBAC) or Claims Based Access Control (CBAC),

**role-based access control** (**RBAC**)[[1]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-2)

is an approach to restricting system access to authorized users. It is used by the majority of enterprises with more than 500 employees,[[3]](https://en.wikipedia.org/wiki/Role-based_access_control" \l "cite_note-autogenerated2010-3) and can implement [mandatory access control](https://en.wikipedia.org/wiki/Mandatory_access_control) (MAC) or [discretionary access control](https://en.wikipedia.org/wiki/Discretionary_access_control) (DAC). RBAC is sometimes referred to as role-based security.

Role-based-access-control (RBAC) is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments. A study by NIST has demonstrated that RBAC addresses many needs of commercial and government organizations. RBAC can be used to facilitate administration of security in large organizations with hundreds of users and thousands of permissions. Although RBAC is different from MAC and DAC access control frameworks, it can enforce these policies without any complication. Its popularity is evident from the fact that many products and businesses are using it directly or indirectly.

Within an organization, [roles](https://en.wikipedia.org/wiki/Role_(computer_science)) are created for various job functions. The permissions to perform certain operations are assigned to specific roles. Members or staff (or other system users) are assigned particular roles, and through those role assignments acquire the computer permissions to perform particular computer-system functions. Since users are not assigned permissions directly, but only acquire them through their role (or roles), management of individual user rights becomes a matter of simply assigning appropriate roles to the user's account; this simplifies common operations, such as adding a user, or changing a user's department.

Three primary rules are defined for RBAC:

1. Role assignment: A subject can exercise a permission only if the subject has selected or been assigned a role.
2. Role authorization: A subject's active role must be authorized for the subject. With rule 1 above, this rule ensures that users can take on only roles for which they are authorized.
3. Permission authorization: A subject can exercise a permission only if the permission is authorized for the subject's active role. With rules 1 and 2, this rule ensures that users can exercise only permissions for which they are authorized.

Additional constraints may be applied as well, and roles can be combined in a [hierarchy](https://en.wikipedia.org/wiki/Hierarchy) where higher-level roles subsume permissions owned by sub-roles.

With the concepts of [role hierarchy](https://en.wikipedia.org/wiki/Role_hierarchy) and constraints, one can control RBAC to create or simulate [lattice-based access control](https://en.wikipedia.org/wiki/Lattice-based_access_control) (LBAC). Thus RBAC can be considered to be a superset of LBAC.

When defining an RBAC model, the following conventions are useful:

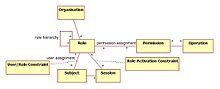
* S = Subject = A person or automated agent
* R = Role = Job function or title which defines an authority level
* P = Permissions = An approval of a mode of access to a resource
* SE = Session = A mapping involving S, R and/or P
* SA = Subject Assignment
* PA = Permission Assignment
* RH = Partially ordered Role Hierarchy. RH can also be written: ≥ (The notation: x ≥ y means that x inherits the permissions of y.)
  + A subject can have multiple roles.
  + A role can have multiple subjects.
  + A role can have many permissions.
  + A permission can be assigned to many roles.
  + An operation can be assigned many permissions.
  + A permission can be assigned to many operations.

A constraint places a restrictive rule on the potential inheritance of permissions from opposing roles, thus it can be used to achieve appropriate [separation of duties](https://en.wikipedia.org/wiki/Separation_of_duties). For example, the same person should not be allowed to both create a login account and to authorize the account creation.

Thus, using [set theory](https://en.wikipedia.org/wiki/Set_theory) [notation](https://en.wikipedia.org/wiki/Mathematical_notation):

* {\displaystyle PA\subseteq P\times R} and is a many to many permission to role assignment relation.
* {\displaystyle SA\subseteq S\times R} and is a many to many subject to role assignment relation.
* {\displaystyle RH\subseteq R\times R}

A subject may have *multiple* simultaneous sessions with different permissions.

[](https://en.wikipedia.org/wiki/File:Role-based_access_control.jpg)

RBAC

**Standardized levels**[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=2" \o "Edit section: Standardized levels)]

*See also:*[*NIST RBAC model*](https://en.wikipedia.org/wiki/NIST_RBAC_model)

The NIST/ANSI/[INCITS](https://en.wikipedia.org/wiki/INCITS) RBAC standard (2004) recognizes three levels of RBAC:[[4]](https://en.wikipedia.org/wiki/Role-based_access_control" \l "cite_note-BelussiCatania2007-4)

1. core RBAC
2. hierarchical RBAC, which adds support for inheritance between roles
3. constrained RBAC, which adds separation of duties

Relation to other models[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=3" \o "Edit section: Relation to other models)]

RBAC is a flexible access control technology whose flexibility allows it to implement [DAC](https://en.wikipedia.org/wiki/Discretionary_access_control)[[5]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-5) or MAC.[[6]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-6) [DAC](https://en.wikipedia.org/wiki/Discretionary_access_control) with groups (e.g., as implemented in POSIX file systems) can emulate RBAC.[[7]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-7) [MAC](https://en.wikipedia.org/wiki/Mandatory_access_control) can simulate RBAC if the role graph is restricted to a tree rather than a [partially ordered set](https://en.wikipedia.org/wiki/Partially_ordered_set).[[8]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-8)

Prior to the development of RBAC, the [Bell-LaPadula](https://en.wikipedia.org/wiki/Bell-LaPadula) (BLP) model was synonymous with MAC and [file system permissions](https://en.wikipedia.org/wiki/File_system_permissions) were synonymous with DAC. These were considered to be the only known models for access control: if a model was not BLP, it was considered to be a DAC model, and vice versa. Research in the late 1990s demonstrated that RBAC falls in neither category.[[9]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-9)[[10]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-10) Unlike [context-based access control](https://en.wikipedia.org/wiki/Context-based_access_control) (CBAC), RBAC does not look at the message context (such as a connection's source). RBAC has also been criticized for leading to role explosion,[[11]](https://en.wikipedia.org/wiki/Role-based_access_control" \l "cite_note-11) a problem in large enterprise systems which require access control of finer granularity than what RBAC can provide as roles are inherently assigned to operations and data types. In resemblance to CBAC, an Entity-Relationship Based Access Control (ERBAC, although the same acronym is also used for modified RBAC systems,[[12]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-12) such as Extended Role-Based Access Control[[13]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-13)) system is able to secure instances of data by considering their association to the executing subject.[[14]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-14)

RBAC differs from [access control lists](https://en.wikipedia.org/wiki/Access_control_lists) (ACLs), used in traditional discretionary access-control systems, in that it assigns permissions to specific operations with meaning in the organization, rather than to low level data objects. For example, an access control list could be used to grant or deny write access to a particular system file, but it would not dictate how that file could be changed. In an RBAC-based system, an operation might be to 'create a credit account' transaction in a financial application or to 'populate a blood sugar level test' record in a medical application. The assignment of permission to perform a particular operation is meaningful, because the operations are granular with meaning within the application. RBAC has been shown to be particularly well suited to separation of duties (SoD) requirements, which ensure that two or more people must be involved in authorizing critical operations. Necessary and sufficient conditions for safety of SoD in RBAC have been analyzed. An underlying principle of SoD is that no individual should be able to effect a breach of security through dual privilege. By extension, no person may hold a role that exercises audit, control or review authority over another, concurrently held role.[[15]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-16)

**Comparing with ACL**[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=4" \o "Edit section: Comparing with ACL)]

An alternative option to the RBAC model is the [ACL model](https://en.wikipedia.org/wiki/Access_control_list). A "minimal RBAC Model", *RBACm*, can be compared with an ACL mechanism, *ACLg*, where only groups are permitted as entries in the ACL. Barkley (1997)[[17]](https://en.wikipedia.org/wiki/Role-based_access_control" \l "cite_note-17) showed that *RBACm* and *ACLg* are equivalent.

In modern [SQL](https://en.wikipedia.org/wiki/SQL) implementations, like [ACL of the](http://book.cakephp.org/2.0/en/core-libraries/components/access-control-lists.html) [CakePHP framework](https://en.wikipedia.org/wiki/CakePHP" \o "CakePHP), ACL also manage groups and inheritance in a hierarchy of groups. So, specific "modern ACL" implementations can be compared with specific "modern RBAC" implementations, better than "old (file system) implementations".

For data interchange, and for "high level comparisons", ACL data can be translated to XACML.

**Attribute based access control**[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=5" \o "Edit section: Attribute based access control)]

Attribute-based access control or [ABAC](https://en.wikipedia.org/wiki/Attribute-based_access_control) is a model which evolves from RBAC to consider additional attributes in addition to roles and groups. In ABAC, it is possible to use attributes of:

* the user e.g. citizenship, clearance,
* the resource e.g. classification, department, owner,
* the action, and
* the context e.g. time, location, IP.

ABAC is policy-based in the sense that it uses policies rather than static permissions to define what is allowed or what is not allowed.

Use and availability[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=6" \o "Edit section: Use and availability)]

The use of RBAC to manage user privileges (computer permissions) within a single system or application is widely accepted as a best practice. A 2010 report prepared for [NIST](https://en.wikipedia.org/wiki/NIST) by the [Research Triangle Institute](https://en.wikipedia.org/wiki/Research_Triangle_Institute) analyzed the economic value of RBAC for enterprises, and estimated benefits per employee from reduced employee downtime, more efficient provisioning, and more efficient access control policy administration.[[3]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-autogenerated2010-3)

In an organization with a heterogeneous IT infrastructure and requirements that span dozens or hundreds of systems and applications, using RBAC to manage sufficient roles and assign adequate role memberships becomes extremely complex without hierarchical creation of roles and privilege assignments.[[18]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-18) Newer systems extend the older [NIST RBAC model](https://en.wikipedia.org/wiki/NIST_RBAC_model)[[19]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-19) to address the limitations of RBAC for enterprise-wide deployments. The NIST model was adopted as a standard by [INCITS](https://en.wikipedia.org/wiki/INCITS) as ANSI/INCITS 359-2004. A discussion of some of the design choices for the NIST model has also been published.[[20]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-20)

RBAC and employees' responsibilities alignment[[edit](https://en.wikipedia.org/w/index.php?title=Role-based_access_control&action=edit&section=7)]

In **Aligning Access Rights to Governance Needs with the Responsibility MetaModel (ReMMo) in the Frame of Enterprise Architecture**[[21]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-21) an expressive Responsibility metamodel has been defined and allows representing the existing responsibilities at the business layer and, thereby, allows engineering the access rights required to perform these responsibilities, at the application layer. A method has been proposed to define the access rights more accurately, considering the alignment of the responsibility and RBAC.[[22]](https://en.wikipedia.org/wiki/Role-based_access_control#cite_note-22)