that the system security policies and mechanisms configured by the systems or security administrators are enforced and tamperproof. Non-discretionary access controls can be installed on many operating systems. Since NDAC does not depend on a subject's compliance with the security policy as in the case of DAC, but is universally applied, it offers a higher degree of protection. Without NDAC, even if a user attempts to comply with well-defined file protection mechanisms, a Trojan horse program could change the protection controls to allow uncontrolled access.

Mandatory Access Control (MAC)

In MAC, access to objects is restricted to subjects based on the sensitivity of the information contained in the objects. The sensitivity is represented by a label. Only subjects that have the appropriate privilege and formal authorization (i.e., clearance) are granted access to the objects. MAC requires sensitivity labels for all the objects and clearance levels for all subjects and access is determined based on matching a subject's clearance level with the object's sensitivity level. Examples of government labels include top secret, secret, confidential, etc. and examples of private sector labels include high confidential, confidential-restricted, for your eyes only, etc.

MAC provides multi-level security since there are multiple levels of sensitivity requirements that can be addressed using this form of access control.

MAC systems are more structured in approach and more rigid in their implementation because they do not leave the access control decision to the owner alone as in the case of DAC, but both the system and the owner are used to determine whether access should be allowed or not. A common implementation of MAC is *rule-based access control*. In rule-based access control, the access decision is based on a list of rules that are created or authorized by system owners who specify the privileges (i.e., read, write, execute, etc.) that the subjects (users) have on the objects (resources). These rules are used to provide the need-to-know level of the subject. Rule-based MAC implementation requires the subject to possess the "need to know" property which is provided by the owner but in addition to the owner deciding who possesses "need to know", in MAC, the system determines access decisions based on clearance and sensitivity.

Role-Based Access Control (RBAC)

Since the mapping of each subject to a resource (as in the case of DAC) or the assignment of subjects to clearance levels and objects to sensitivity levels (as in the case of MAC) can be an arduous task, for purposes of ease of user management,

a more agile and efficient access control model is role based access control (RBAC). Roles are defined by job function which can be used for authorization decisions. Roles define the trust levels of entities to perform desired operations. These roles may be user roles or service roles. In RBAC, individuals (subjects) have access to a resource (object) based on their assigned role. Permissions to operate on objects such Create, Read, Update or Delete are also defined and determined based on responsibilities and authority (permissions) within the job function.

Access that is granted to subjects is based on roles. What this mainly provides is that the resource is not directly mapped to the individual but only to the role. Since individuals can change over time, while roles generally don't, individuals can be easily assigned to or revoked from roles, thereby allowing ease of user management. Roles are then allowed operations against the resource as depicted in *Figure 2.7*.

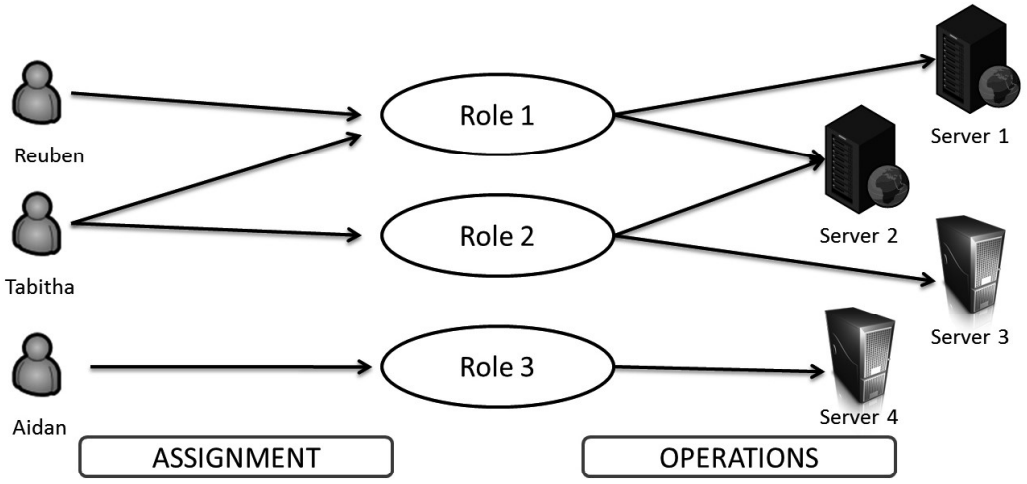

Figure 2.7 – **Role Based Access Control (RBAC)**

RBAC can be used to implement all the three types of access control models i.e., DAC, NDAC and MAC. The discretionary aspect is that the owners need to determine which subjects need to be granted what role. The non-discretionary aspect is that the security policy is universally enforced on the role irrespective of the subject. It is also a form of MAC where the role is loosely analogous to the process of clearance levels (granting memberships) and the objects requested are labeled (associated operational sensitivities), but RBAC is not based on multi-level security requirements.

RBAC in Relation to Least Privilege and Separation of Duties

Roles support the principle of least privilege, since roles are given just the needed privileges to undertake an operation against a resource. When RBAC is used for authorization decisions, it is imperative to ensure that the principle of Separation of Duties (SoD) is maintained. This means that no individual can be assigned to two roles that are mutually exclusive in their permissions to perform operations. For example, a user should not be in a datareader role as well as a more privileged database owner role at the same time. When users are prevented from being assigned to conflicting roles, then it is referred to as static SoD. Another example that demonstrates SoD in RBAC is that a user who is in the auditor role cannot also be in the teller role at the same time. When users are prevented from operating on resources with conflicting roles then it is referred to as dynamic SoD.

RBAC implementations require explicit "role engineering" to determine roles, authorizations, role-hierarchies and constraints. The real benefit of RBAC over other access control methods includes the following:

- Simplified subjects and objects access rights administration
- Ability to represent the organizational structure
- Force enterprise compliance with control policies more easily and effectively.

Role Hierarchies

Roles can be hierarchically organized and when such a parent-child tree structure is in place, it is commonly referred to as a *role hierarchy.* Role hierarchies define the inherent relationships between roles. For example, an admin user may have read, write and execute privileges, while a general user may have just read and write privileges and a guest user may only have read privilege. In such a situation, the guest user role is a subset of the general user role which in turn is a subset of the admin user role as illustrated in *Figure 2.8*

In generating role hierarchies, we start with the most common and least privileged permissions (e.g., read) for all users and then iterate permissions to be more restrictive (e.g., write, execute), assigning them to roles (guest, user, administrator) which are then assigned to users. When determining role hierarchy it is also important to identify contextual and content based constraints and grant access rights based on "*only if*" or "*if and only if*" relationships. Just basing the access decisions on an *if* relationships does not provide real separation of

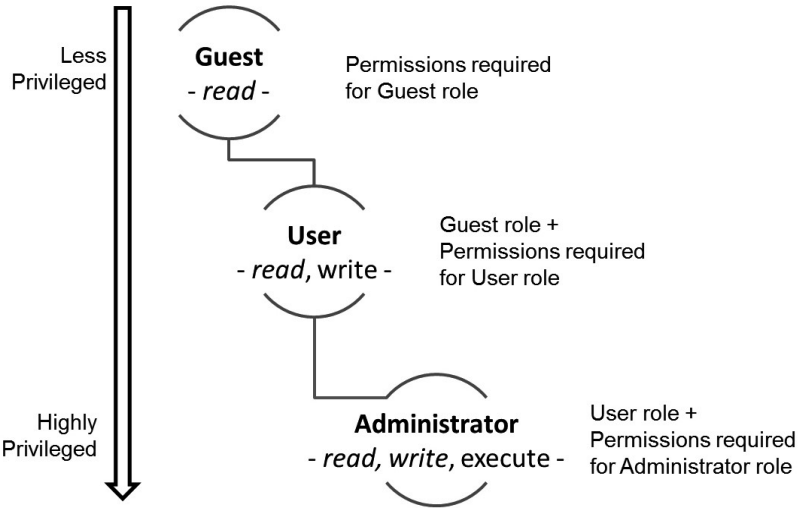

Figure 2.8 – **Role Hierarchy**

duties. For example a doctor should be allowed to view the records of a patient, *only if* or *if and only if* that patient whose records are requested for is assigned to the doctor, not just *if* the requestor is a doctor.

Roles and Groups

Although it may seem like there is a high degree of similarity between roles and groups, there is a distinction that make RBAC more preferable for security than groups. A group is a collection of users and not a collection of permissions. In a group, permissions can be assigned to both users and groups to which users are part of. The ability to associate a user directly with permissions in group-based access control can be the Achilles heel for circumventing access control checks, besides making it more difficult to manage users and permissions. RBAC mandates that all access is done only through roles and permissions are never directly assigned to the users but to the roles and this addresses the challenges that one can have with group-based access control mechanisms.

Resource-Based Access Control

When the list of all users of your software are not known in advance, as in the case of a distributed Internet application, then DAC, and MAC implementation using subject (user) mapping to objects (resources) may not always be possible. In such situations, access can also be granted based on the resources. Resource based access control models are useful in architectures that are distributed and multi-tiered including service oriented architectures. Resource based access control models can be broadly divided into