



Technical White Paper
*Implementing Role-Based
Access Controls in the
Enterprise*

*Toby Emden,
Practice Director – Identity Management
Qubera Solutions, Inc.*

Version: 1.2 - DRAFT
Date: February 21, 2011

1. Abstract

As information systems become increasingly complex and regulatory requirements place a mounting burden on organizations, entitlements management is rapidly becoming a critical business priority rather than merely a function of the I.T. department. This paradigm shift requires organizations to reevaluate traditional approaches to entitlements management, where processes are often decentralized, inconsistent and can often result in audit/compliance violations.

The operational costs and organizational risks incurred by such processes create an urgent challenge for the enterprise. This has led to the increasing adoption of Role Based Access Controls (RBAC) as a holistic solution for entitlements management.

This document aims to provide a technology agnostic overview of the drivers, definitions, planning considerations and recommended implementation strategies for RBAC. It expands upon the RBAC reference model published by the National Institute of Standards and Technology (NIST) and the authoritative ANSI/INCITS 359-2004 standard.

The target audience for this document includes information architects, business technology leaders and security professionals.

2. Introduction

In a traditional security administration model, entitlements management is highly decentralized. An “object” such as a database, application or file contains a set of permissions that may be granted to an authorized “subject” such as a user or group, conferring upon it the privilege to perform certain actions. The underlying assumption of this model is that each object has an owner who grants permissions, typically by adding subjects to an Access Control List (ACL).

Many modern systems provide an additional level of abstraction with groups to which subjects may belong. A group is merely another form of subject that can be added to an ACL, thus conferring implicit privileges upon group members.

Because the object owner has discretion over permission assignments, this type of security administration model is known as Discretionary Access Control (DAC).

2.1 Example of Discretionary Access Controls

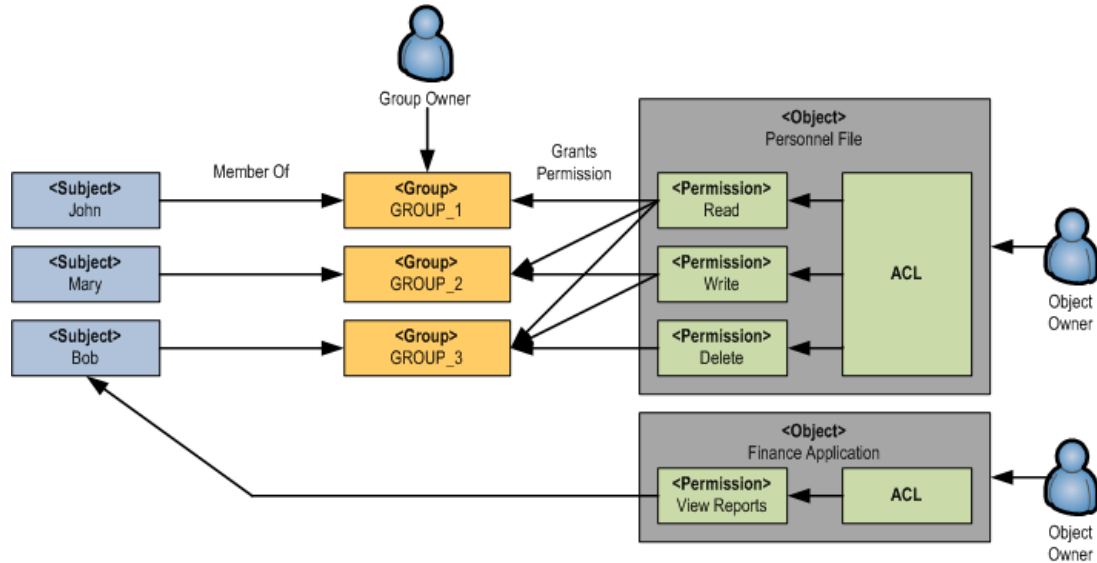


Figure 1: Example of Discretionary Access Controls

Figure 1 illustrates how privileges are typically granted in a DAC-based model. In this diagram, the subject Bob belongs to GROUP_3. By virtue of that group being granted the appropriate privileges in an object ACL, he is permitted to read, write and delete a personnel file. Bob also owns a privilege to view reports in a finance application, although this permission is explicitly assigned to him, rather than by virtue of a group.

Mary, meanwhile, belongs to GROUP_2, which only has read/write permissions on the personnel file. The owner of GROUP_2 added Mary to the group without consulting the owner of the personnel file, who works in a different part of the organization.

When this model is extrapolated to every system and user in a large enterprise, the number of individual entitlements can run into the millions, and even in this simple example, some risks become immediately apparent.

2.2 Shortcomings of Discretionary Access Controls

In the above example, the object owner may have no knowledge of John, Mary or Bob, much less whether their job functions require them to have a specific privilege. The functional disconnect between object owner and subject is complicated by the existence of abstract groups, since the object owner may have no control over group memberships; a challenge that is compounded by systems that support nested groups, such as Active Directory.

In theory, such issues can be addressed by assigning permissions directly to a subject, as with Bob's privilege to view reports in the finance application in Figure 1. However, this does not address the possibility of an object owner being unaware of the privilege needs for a particular job function. In fact, one of the major incentives for using groups is to abstract risk analysis from the object owner, who would otherwise be responsible for granting permissions to many individual subjects.

Due to the abstraction offered by groups, it is quite common to hear administrators inaccurately describe them as "roles", especially in large corporate directories such as Active Directory or RACF. Groups, however, are merely collections of subjects, whereas a role is an aggregation of entitlements. In other words, a role describes exactly what permissions are conferred upon its users, while groups do not. Thus, group owners typically have no knowledge of how their groups are being used by object owners to assign permissions. The use of groups places accountability upon group owners, who have discretion over group memberships but often have no control over what members are permitted to do.

In organizations with large enterprise directories that may contain thousands of groups, group ownership is often ambiguous and inconsistent. This is particularly true if policies governing group creation, ownership, modification and review, are poorly observed. It is not unusual to hear of large enterprises where tens of thousands of groups have been created, with no record of who created them, who owns them or how applications or using them to authorize users. Because of this uncertainty, group clean-up can be a challenge; nobody wants to risk breaking applications by deleting groups that may be in use.

In DAC-based systems, object owners are often I.T. employees who possess limited knowledge of corporate governance or compliance issues, and are thus ill equipped to quantify the potential risk of assigning permissions to a particular user or group.

In summary, discretionary access controls possess several major shortcomings from a security and compliance perspective:

- 1) Failure to link entitlements to a user's business function, leading to potential segregation of duty (SoD) violations. Theoretically, there is nothing to prevent an employee who has the authority to approve an invoice from also having the authority to pay that same invoice, particularly if those operations exist on different systems, or if privileges to perform those operations are granted by different individuals.
- 2) A group is merely a collection of users, whereas a role is an aggregation of entitlements. Even though object owners may have control over which groups are permitted to access an object, they may not always know which individuals belong to those groups, not to mention whether the implicit permissions granted to any group member might result in a policy violation.

- 3) Conversely, group owners may not be able to determine how their groups are being used to provide authorizations. There is nothing to prevent an object owner from granting permissions to a pre-existing group and inadvertently creating a breach without the group owner ever being aware of the fact.
- 4) In practice, entitlements tend to be cumulative, as DAC processes rarely account for access revocation when a user's job function changes. This can result in users accruing more privileges than they need, in addition to the risk of "orphaned" accounts lingering in application repositories after a termination event
- 5) Objects tend to be I.T. assets such as applications, systems and data. Because DAC assumes the existence of object owners, it promotes the idea of entitlements management as an I.T. function. I.T. employees generally have little knowledge of what privileges are required to perform a business function. This results in permissions being granted on the basis of trust (for instance, manager or supervisor approval) rather than risk assessment.

All of these factors expose businesses to potentially catastrophic regulatory violations and security breaches. One of the most notorious real-world examples occurred in 2008, when a trader at French bank Societe Generale exploited inadequate access controls to execute a series of unauthorized trades, resulting in losses of \$7.2 billion to his employer. Although breaches of this magnitude are rare, even minor violations can incur substantial cost to an enterprise in terms of brand damage, material losses, legal ramifications and regulatory fines.

2.3 The Hidden Costs of DAC

As the economy becomes ever more dependent upon the integrity of mission-critical information systems, businesses face increasingly stringent regulatory mandates, such as SOX, GLBA, HIPAA and PCI-DSS. It is reasonable to expect that regulatory oversight of access governance will intensify over the coming years, especially with the continued proliferation of cloud based solutions.

The increasing complexity of information systems, coupled with strict compliance mandates, can result in significant labor costs for businesses that rely on discretionary access controls. Such costs are reflected in the person hours consumed by manual certifications, attestations, remediations and production of compliance reports. This is in addition to the obvious costs of manual provisioning and lost productivity incurred by users waiting for discretionary access to be granted.

Another hidden cost of DAC involves the number of superfluous software licenses that tend to exist in larger organizations. As mentioned previously, users tend to accrue entitlements over time. This can result in unnecessary costs if licenses aren't recovered once employees no longer require them.

Thus for many organizations, entitlements management is rapidly becoming a core business imperative as opposed to an I.T. back office function. This has created an urgent need for the enterprise to streamline and automate security administration processes, and provide for more robust enforcement of audit and regulatory requirements.

3. RBAC Overview and Principles

Role Based Access Controls restrict access to sensitive information assets by aggregating entitlements in a form that has meaning to the business. High-level business roles are aligned to organizational structures and job functions, while fine-grained technical roles aggregate system level entitlements. This approach to entitlements management is policy-based rather than discretionary.

RBAC is based on the principle of least privileged access; a role only grants the minimum privileges required by an individual to perform their job. It is therefore a variant of Mandatory Access Control (MAC), albeit with a more sophisticated approach to organizing entitlements. Furthermore, roles support concepts such as inheritance, hierarchy, cardinality, dynamic and static separation, all of which distinguish them from user groups. Additional features beginning to appear in many recent IAM products, such as role attributes, attribute aware roles, rule-driven user/role assignments, contextually aware roles, and conditional inheritance, have enhanced the ability of RBAC to deliver business value.

Whereas groups are merely collections of named users, a role is an aggregation of entitlements. The basic principle of RBAC is that a subject may execute a transaction only if the subject has been assigned a role that has the requisite permissions to do so. Roles can be statically or dynamically assigned to a user, while role hierarchies simplify role modeling by enabling senior roles to inherit the entitlements of more junior roles. Constraints can be applied to role assignments and hierarchies to prevent conflicting roles from being assigned to a user, and to prevent users from acquiring too much authority. The application of role constraints is widely accepted as an effective mechanism for preventing SoD violations.

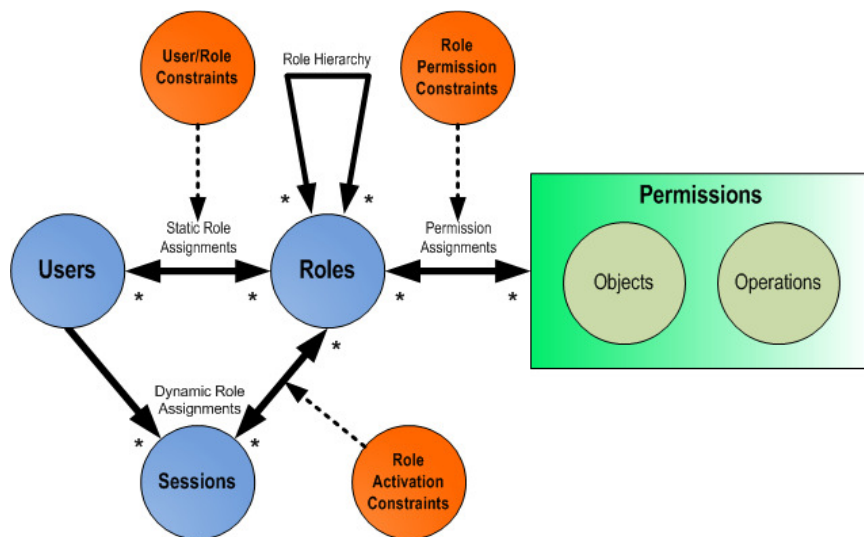


Figure 2: Standard Role Architecture

3.1 Role Types

Strictly speaking, the ANSI/INCITS 359-2004 standard does not distinguish between different types of role, nor does it make any assumptions about how they should be assigned to users. However, it is strongly recommended that an RBAC design should make distinctions between different role types in order to simplify ownership and maintenance issues.

In general, there are two major types of role that should constitute an RBAC design.

3.1.1 Business Roles

A business role is often described interchangeably as an “enterprise”, “functional” or “organizational” role. It provides an abstract set of permissions for users who share a common business classification, organization or job function. Some RBAC designs make distinctions between enterprise, organizational and functional roles, as in the following example:

Enterprise Roles: EMPLOYEE, CONTRACTOR, VENDOR
Organizational Roles: FINANCE, SALES, HR, EUROPE, ASIA, NORTH AMERICA
Functional Roles: PROJECT MANAGER, CASHIER, SALES EXECUTIVE

A user may have several business roles, which can be assigned statically, dynamically or by using inheritance. In Figure 3, Bob is assigned the SALES EXECUTIVE functional role, which inherits permissions from the SALES and NORTH AMERICA organizational roles, and finally from the EMPLOYEE enterprise role. The use of general inheritance enables Bob to acquire all the permissions in this hierarchy by virtue of a single role assignment.

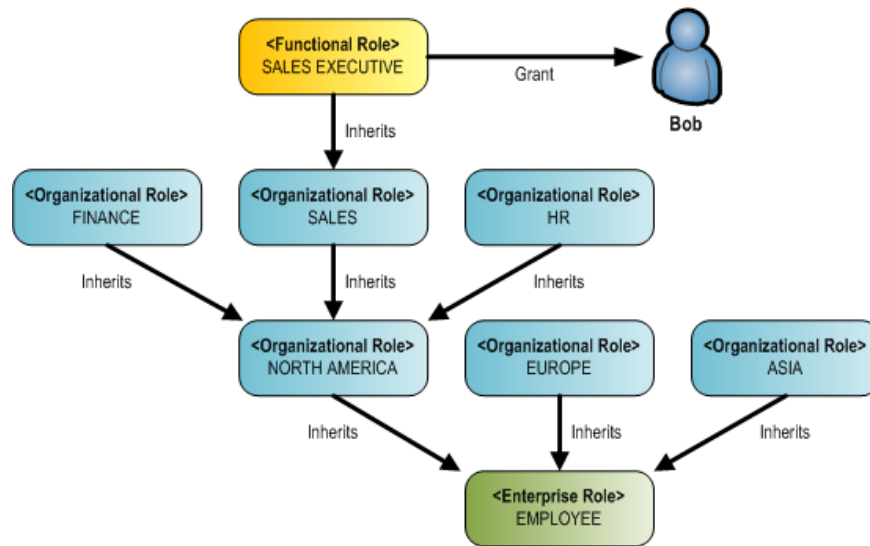


Figure 3: Example of a Business Role Hierarchy using General Inheritance

Some organizations may find a general inheritance pattern like this too dangerous, as it implies the cumulative aggregation of privileges. This would result in the company owner acquiring every privilege in his organization. No matter how senior or trusted an employee might be, allowing privileges to propagate upwards in such an uncontrolled fashion could expose the enterprise to SoD violations, security breaches or even malicious code exploits. Furthermore, it violates the guiding principle of only granting the minimum privileges that an employee requires to perform their job; for instance, the VP of Engineering would probably never require root access to a production server, even if one or more of his subordinates had that privilege.

One alternative to general inheritance is to use a flat RBAC pattern, in which Bob is directly assigned multiple business roles.

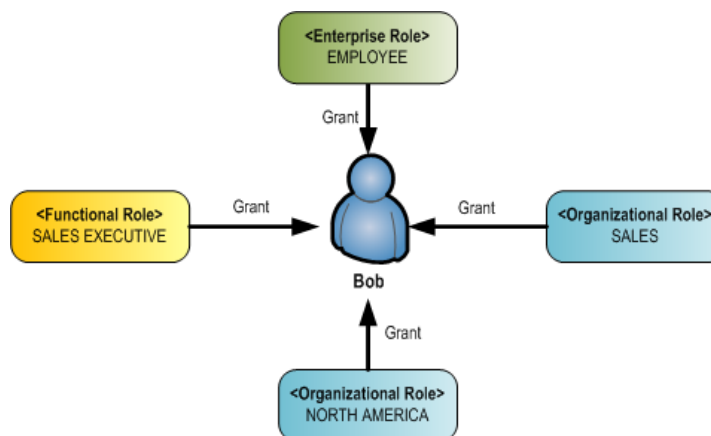


Figure 4: Example of Flat RBAC

From Bob’s perspective, the approach shown in Figure 4 would achieve the same end result as that in Figure 3. The distinction is mainly operational. In a general inheritance hierarchy, revoking the SALES EXECUTIVE role would by extension also remove Bob from the SALES, NORTH AMERICA and EMPLOYEE roles. This may not be a desirable outcome, especially if Bob were only transferring to another job function within the same department. Using flat RBAC, on the other hand, each role would have to be explicitly revoked, which implies significant operational burden if, for example, Bob were transferred to another continent, requiring separate revocation of three roles.

To ensure maximum flexibility in role designs and prevent the uncontrolled aggregation of privileges, a hybrid of general inheritance and flat RBAC is recommended. This pattern is known as a limited inheritance hierarchy, and is explained in greater detail in section 3.3.2.

There are no absolutes when it comes to modeling business roles, since every organization has different hierarchies, needs and policies. When weighing the merits of general inheritance, limited inheritance and flat RBAC patterns, role designers must consider various factors. These include specific business use cases that predicate against inheritance, separation of duty constraints, the operational burden of user/role assignments in a flat RBAC or limited inheritance model, organizational turnover, matrixed reporting hierarchies, IAM product features and the frequency of privilege changes within specific business units.

In many cases, RBAC designs begin with a series of flat role assignments and evolve over time into more sophisticated inheritance hierarchies as the relationships between roles become more clearly defined. The topic of role service levels will be discussed in section 3.3.

3.1.2 Technical Roles

A technical role, often referred to as an “application” or “I.T.” role, contains a set of entitlements to systems, sub-systems, objects and operations. Although RBAC conventions do not expressly prohibit the direct assignment of technical roles to users, it is recommended that technical roles should always be contained by business roles and should never be directly assigned to a user.

Technical roles may contain both coarse-grained permissions for entire sub-systems and fine-grained permissions to individual objects and operations. They can be mapped to business roles at any level of an organizational hierarchy, and they support inheritance in the same way as business roles, with the caveat that technical roles can only inherit the permissions of other technical roles.

Figure 5 extends the general inheritance example used in section 3.1.1 to illustrate how business roles encapsulate the permissions of technical roles.

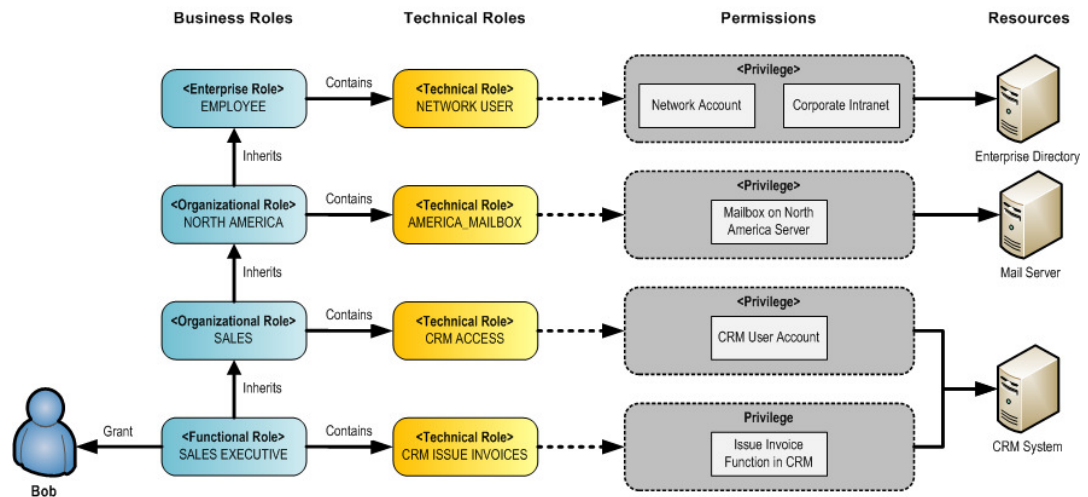


Figure 5: Mapping of Technical Roles to Business Roles

As in the earlier example, Bob is assigned the business role of SALES EXECUTIVE, which inherits from SALES, NORTH AMERICA and EMPLOYEE respectively. Each of these business roles contains one or more technical roles that describe either baseline systems access, or an operation that role members are authorized to perform.

Although there is no functional difference between inheritance and containment, as SALES EXECUTIVE could be said to inherit the entitlements of CRM ISSUE INVOICES, the semantic distinction is used to highlight the relationship between a business and technical role. Technical roles may inherit from each other, but may never inherit from a business role. As a best practice, technical roles should never be assigned directly to users, although the ANSI/INCITS standard does not explicitly prohibit this from occurring, and there may be specific exceptions where this needs to occur.

3.1.3 Summary

Maintaining a clear distinction between business and technical roles is integral to sound role design. It helps to establish unambiguous spheres of ownership in which business roles are owned by the business and technical roles are owned by I.T. staff. Enforcing the rule of only allowing business roles to be assigned directly to a user addresses many of the challenges with DAC models discussed earlier.

To conclude, the key capabilities of technical and business roles can be defined as follows:

	Business Roles	Technical Roles
May be assigned directly to a user	Yes	No
May contain business roles	Yes	No
May contain technical roles	Yes	Yes
May contain permissions on objects and operations	No	Yes
Support for role constraints	Yes	Yes

3.2 Automating Role Assignments

RBAC designs often overlook the operational aspects of assigning roles to users. Even without automation of user/role assignments, RBAC delivers tangible benefits in the form of streamlined entitlements management, policy enforcement and regulatory compliance. But the true value of RBAC is realized when role assignments are automated.

A standard approach for automating roles is to use an enterprise user provisioning system. Typically, the provisioning system will already contain a series of identity attributes for every user, derived from a system of record such as an HR database. As roles are discovered during the role modeling process, the attributes of role candidates can be analyzed to identify common denominators that are used to generate membership rules. Membership rules can be incorporated into the provisioning engine so that roles are automatically assigned on the basis of user attributes.

Consider the example shown in the section 3.1.2, in which every active employee received a technical role of NETWORK USER and thus acquired privileges to access the network and corporate intranet. This is an extremely basic role that would likely be discovered in the early stages of a role project. Assuming the existence of a user provisioning system, the high-level steps for automating assignment of that role are as follows:

- a) Configure the provisioning system to manage accounts in the enterprise directory if it is not already doing so, and ensure that accounts in the enterprise directory are linked to the correct HR records.
- b) Create a business role of EMPLOYEE and a technical role of NETWORK USER, and link the two, so that EMPLOYEE contains NETWORK USER.
- c) Configure the NETWORK USER role to create an account in the enterprise directory for each role member, and grant permissions to access corporate intranet.
- d) Create a rule within the provisioning system to evaluate the “user type” and “status” attributes for every known identity. If the user type is “Employee” and the user is “Active”, then assign the EMPLOYEE business role to the user.
- e) Set up an ongoing synchronization process to evaluate this rule every time an HR record changes. This will ensure that new hires automatically receive the EMPLOYEE role. Conversely, if a user is terminated, they will be automatically removed from the role, which in turn should revoke their privileges.

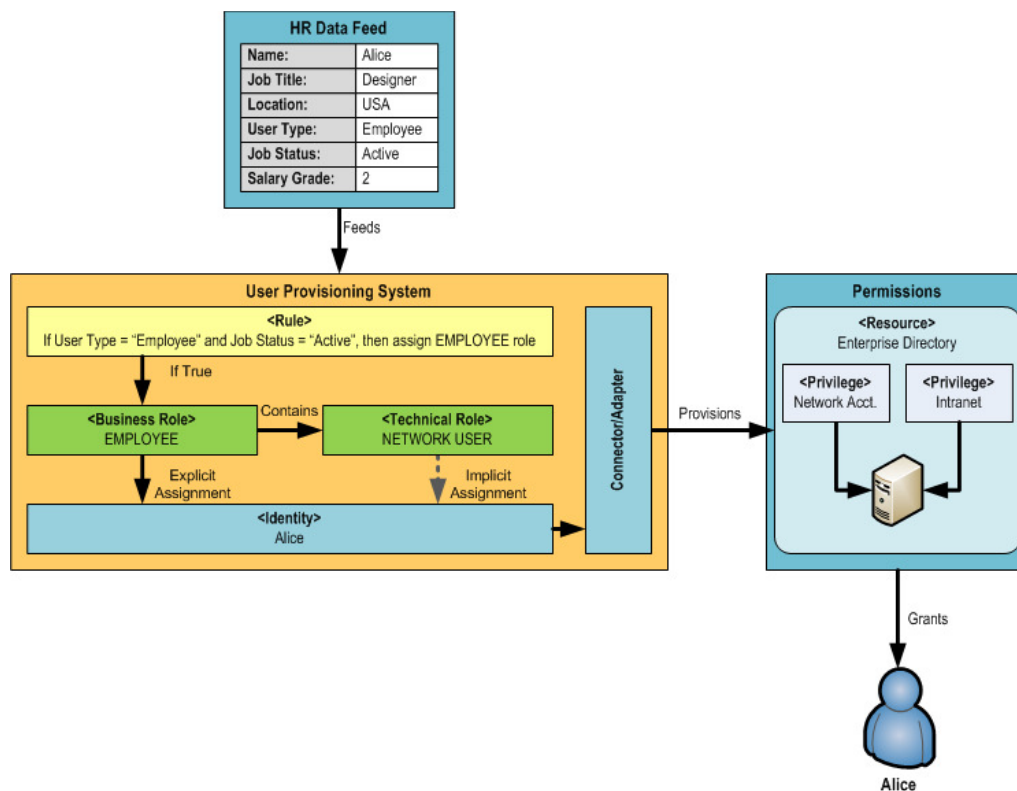


Figure 6: Example of Rule Driven RBAC

In this use case, Alice’s identity attributes are evaluated by a rule in the user provisioning system to determine whether to assign her the EMPLOYEE business role. If the condition is true, then the role is assigned, and the provisioning system automatically grants the appropriate privileges. Alice is therefore provided with access to the network and corporate intranet by virtue of her implicit membership of the NETWORK USER technical role.

It is thus possible to define a set of rules that can automate role assignments and enforce restrictions on them; users who fail to meet the condition(s) associated with each role could be prevented from being assigned to it.

Business Role	Condition to Evaluate
EMPLOYEE	User Type = “Employee” and Job Status = “Active”
CONTRACTOR	User Type = “Contractor” and Job Status = “Active”
NORTH AMERICA	Location = “USA”
SALES	Department = “Sales”
SALES EXECUTIVE	Job Title = “Sales Exec” and Salary Grade > 3

The advantage of this approach is that if Alice were terminated—in which case her job status attribute would no longer be “Active”—the provisioning system could automatically unassign the EMPLOYEE business role, and by extension, the privileges granted by the NETWORK USER technical role. This addresses three major issues with discretionary access controls; namely, the overhead of manual access administration, the accumulation of privileges, and the risk of orphaned accounts lingering after an employee has been terminated.

Order of precedence is an important concept when designing rules, especially in an inheritance hierarchy. If Alice were a sales executive, for example, then assigning the SALES EXECUTIVE role would be sufficient to provide her with the inherited privileges of more junior roles. This precludes the need to evaluate the EMPLOYEE rule. Depending on the identity management tools being used, revoking the SALES EXECUTIVE role may not automatically remove Alice from the EMPLOYEE role if she had not acquired it through her subsequent assignment to the SALES EXECUTIVE role. This may be a desirable outcome, but is nevertheless a design factor that requires careful consideration when developing membership rules.

Understanding the rules and policies that govern user/role assignments is just as important as establishing what entitlements a role should aggregate. Most modern user provisioning systems support the use of rule-driven user/role assignments, and role analytics tools can assist rule design by identifying common identity attributes within a population of role candidates. Therefore, it is strongly recommended that RBAC requirements are considered in the early stages of implementing an enterprise user provisioning system.

3.2.1 Validating Membership Rules

Careful consideration needs to be applied when creating rules, as it is easy to create rules that not only assign a role to the correct users, but also assign it to those who should not have it. This violates RBAC’s basic premise of least privilege access.

Figure 7 illustrates a poorly conceived rule that is used to evaluate a target population of five users:

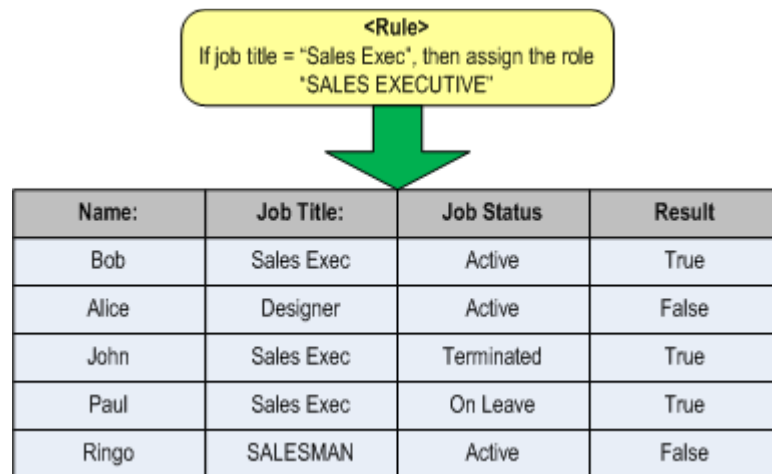


Figure 7: Incorrect Rule Evaluation

The rule has assigned the EMPLOYEE role to Bob, John and Paul. However, John has been terminated and Paul is on leave, so they are false positives, as neither of them should be assigned to the role. Ringo, meanwhile, is a false negative, as he should receive it but has been overlooked by the rule because his HR record has an incorrect job title.

Even if Ringo’s HR record were corrected to reflect the correct job title, the rule itself still needs refinement in order to prevent unauthorized users such as John and Paul from being incorrectly assigned.

The corrected rule is illustrated in Figure 8:

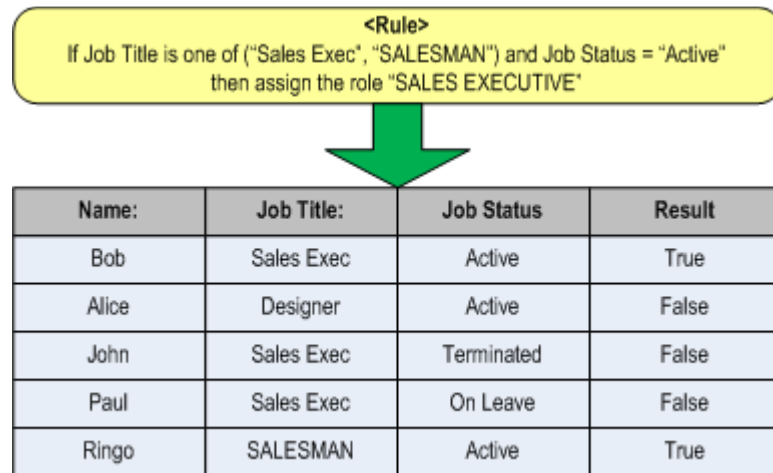


Figure 8: Correct Rule Evaluation

When designing membership rules, it is important to ensure that they only identify users who should be assigned to a role, and not those who shouldn't. It is therefore recommended that every rule should be subject to a review and validation process so users aren't inadvertently assigned more privileges than they need.

3.2.2 Attribute-Aware Roles

Security relevant user attributes, such as those derived from an HR database, are a key component of automating role assignments. However, roles themselves should also be attribute-aware. This allows for the dynamic assignment of attribute values within target systems based on non-predictable criteria.

Consider the example of a retail chain that has stores in five locations. Sales assistants in all five locations are granted a SALES_ASSISTANT role, which grants them access to a POS system. But fine-grained system entitlements within the POS system only allow them to access it during one of three assigned shifts, and they are restricted to registering sales only for the product lines they have been trained to sell. There are up to five possible product lines for which a user could be authorized, and each user could have any combination of the five.

In this scenario, there are three security relevant attributes (location, shift number and product line). Given that there are five possible locations (L), each with three possible shifts (S) and up to five possible product lines (P), the possible permutations of security aware attributes just for the SALES_ASSISTANT role could be calculated thus:

$$LS(2^P - 1) = 465$$

One way to resolve this problem is to create a unique role for each possible permutation of security relevant attributes. But the creation of 465 roles for a single job function clearly isn't a scalable approach, as it would lead to an explosion in the number of roles, especially if the company decided to open more stores or expand to new product lines. If the same approach were adopted for other functions within the same company, the number of roles would quickly become unmanageable.

The use of attribute-aware roles precludes this scenario by relying on business rules or external input to populate security relevant attributes independently of a static role definition. In the above example, an attribute-aware SALES_ASSISTANT role would grant coarse grained privileges to the POS system, but additional security relevant attributes would be assigned independently, either by evaluating the user’s identity attributes, or by requiring input from a designated member of staff such as a store manager. Once assigned, those attributes would form part of the user’s role assignment.

3.2.3 Conditional Inheritance

A feature that has recently appeared in some identity management products is conditional inheritance, where one role only inherits the permissions of another if certain conditions are met, such as an evaluation of a role member’s attributes. This is a particularly powerful feature that can be used to simplify role designs and prevent unrestricted inheritance of permissions in a general hierarchy. Conditional inheritance is a key feature of EDAC (Enterprise Dynamic Access Control), the U.S. Navy’s extended implementation of the ANSI/INCITS RBAC standard.

Conditional inheritance can be used, for example, to inherit roles based on variables such as a user’s start date, as illustrated in Figure 9:

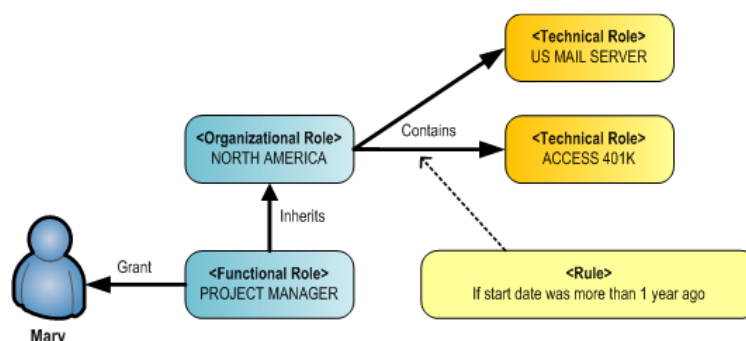


Figure 9: Conditional Inheritance

In this case, Mary has the PROJECT MANAGER role, which inherits from NORTH AMERICA. However, the enterprise policy is that employees may not access the 401K system until they have been in service for one year. Under normal circumstances, Mary would inherit a privilege to access the 401K system by virtue of her implicit assignment of the NORTH AMERICA role. Using conditional inheritance, a rule can be created to prevent inheritance of that privilege simply by evaluating the start date in her HR record. She will, however, still receive any other privileges conferred by the NORTH AMERICA role, such as access to a mail server.

Conditional inheritance offers a powerful alternative to a limited inheritance hierarchy, which is the standard approach for applying inheritance constraints. Limited inheritance is discussed in section 3.3.2.

3.2.4 Role Attributes

Another feature of many recent IAM products is the ability of roles themselves to store attributes. Attributes may contain metadata about the role, such as role owner(s) and

approver(s), last recertification date, creation date, a user friendly description of the role's purpose, data classification and service level.

While role attributes may not directly assist the automation of user/role assignments, this feature is extremely useful for building role catalogs and performing role reviews.

3.3 RBAC Service Levels

The ANSI/INCITS standard identifies four major service levels for roles. Understanding these standards is important when defining a role catalog, which associates each role with permissions, constraints and hierarchies. These standards are defined as follows:

3.3.1 Level 1 – Flat RBAC

Establishes and maintains many-to-many relationships among user-role and permission-role assignments.

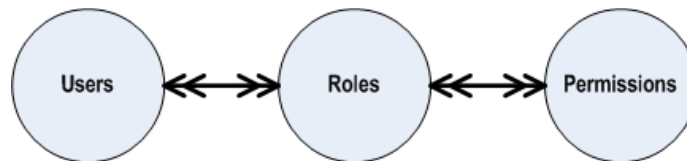


Figure 10: Flat RBAC

Flat RBAC is the most basic design pattern, describing mandatory requirements for any role design in which users acquire permissions by virtue of their role assignments. It assumes that a many-to-many relationship exists between users and roles, and between roles and permissions.

Although Flat RBAC makes no assumptions about contextual or rule-based user/role assignments, it does not specifically exclude this possibility. Nevertheless, Flat RBAC is generally assumed to imply static user/role assignments only.

3.3.2 Level 2 – Hierarchical RBAC

Extends coverage to include user-role and inherited (role-to-role) assignments.

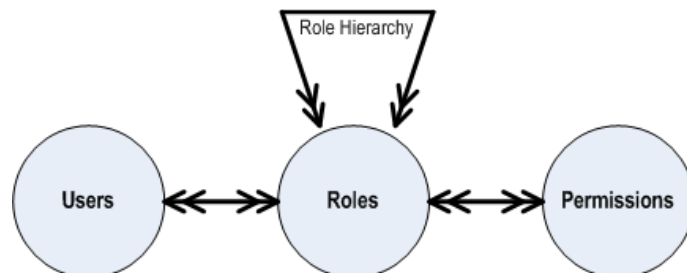


Figure 11: Hierarchical RBAC

Hierarchies introduce the concepts of sharing and aggregation, as illustrated in Figure 12.

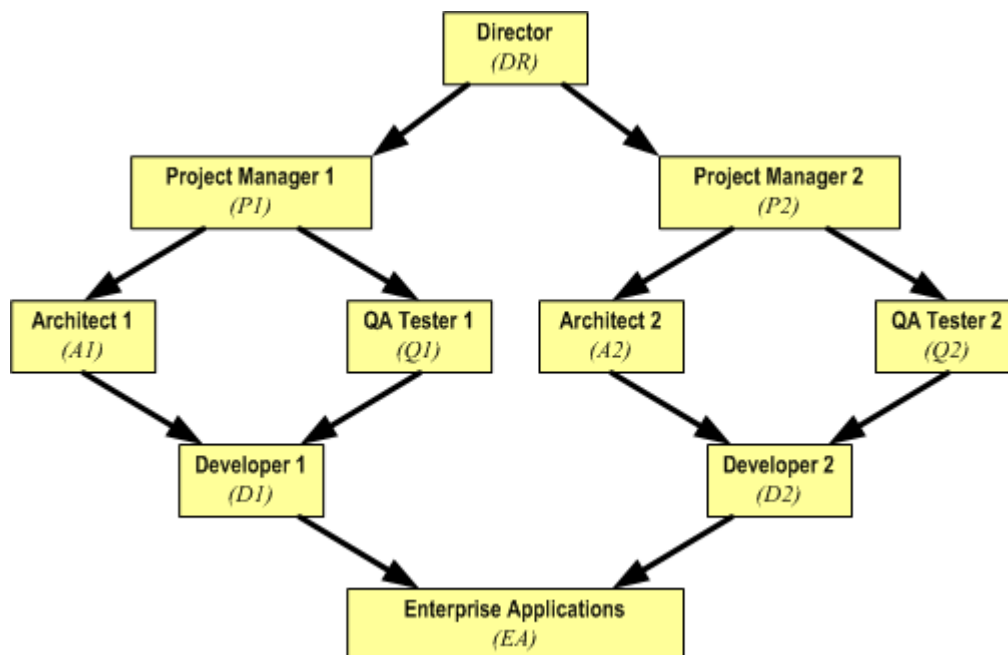


Figure 12: Example of General Inheritance Hierarchy

This example represents an aggregation of roles within a single department, in which two concurrent projects are being managed. The most junior role, *EA*, provides a basic set of privileges that are shared by all employees within the department. This role is inherited by *D(n)* within the context of each project. In turn, the permissions of each developer are inherited by the roles *A(n)* and *Q(n)*, both of which roll up to the appropriate *P(n)* roles. Finally, the *DR* role aggregates all of the junior roles in the department by inheriting from *P(n)*. This project structure is clearly extensible to *n* number of projects.

While this type of general inheritance hierarchy might appear to make logical sense, it can be extremely dangerous in practice, as it may concentrate too much authority in the hands of senior staff, leading to potential SoD violations. For instance, it probably wouldn't be a good idea for the VP of Finance to have the ability to issue and pay the same invoice, or for the Director of Engineering to inherit root permissions to production servers. Furthermore, a general inheritance hierarchy fails to account for SoD constraints, as constraints are always inherited upwards.

Therefore, general inheritance hierarchies tend to be impractical at the enterprise level. Conditional inheritance, which was covered in section 3.2.3, offers a viable option to enforce inheritance constraints, but it is not a feature of the ANSI/INCITS RBAC standard and has only recently appeared as a feature in some IAM tools. The standard approach for isolating sensitive privileges from a general hierarchy is to use limited inheritance, as illustrated in Figure 13.

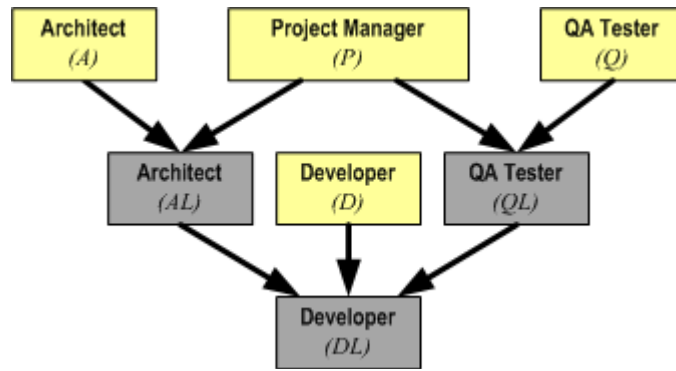


Figure 13: Example of Limited Inheritance Hierarchy

In this example of a limited inheritance hierarchy, the roles *P*, *Q* and *A* inherit the privileges of *AL* and *QL*, which are mere placeholders containing a limited set of entitlements deemed safe for inheritance. The true Architect and QA Tester roles (*A* and *Q*) inherit from their limited counterparts, but are themselves non-inheritable. Likewise, *DL* contains a limited set of developer privileges that are inherited by *AL*, *QL*, and the real developer role, *D*, which exists outside of the general hierarchy. In a limited inheritance hierarchy such as this, *D* would be directly assigned to a developer, *Q* to a QA Tester, and *A* to an Architect. Their limited counterparts would never be directly assigned to users. This approach keeps the general hierarchy intact without allowing sensitive or conflicting privileges to propagate up the inheritance chain.

It is important to note that RBAC makes no assumptions about the existence of a senior or junior role in any role hierarchy. Roles can be assigned independently of each other and can exist outside of a general inheritance hierarchy. It should also be noted that the conditional inheritance feature, now supported by more sophisticated IAM tools, offers a viable alternative to limited inheritance, with the additional benefit of not requiring placeholder roles to be created.

3.3.3 Level 3 – Constrained RBAC

Introduces separation of duty constraints and policies, both static and dynamic.

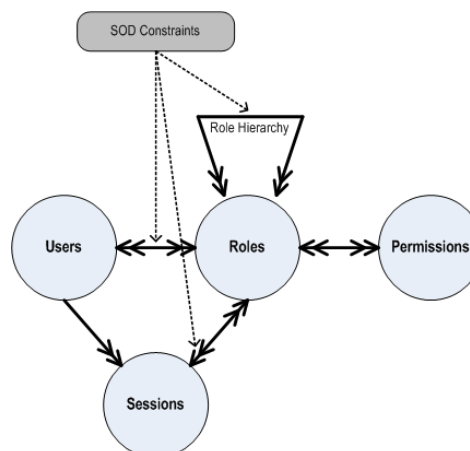


Figure 14: Constrained RBAC

Separation of duty ensures that the same user cannot have conflicting roles that would provide them with an unacceptable level of authority. Constraints can be applied to user/role assignments (static constraints), to session/role assignments (dynamic constraints), or to role hierarchies.

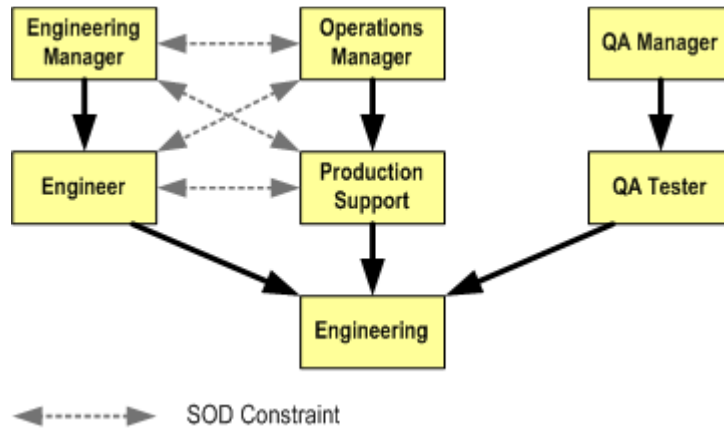


Figure 15: Inherited Constraints

Constraints are inherited, as shown in Figure 15. In this example, an SoD constraint ensures that no user can be assigned to both the Engineer and Production Support roles. Therefore, the Operations Manager role, which contains Production Support, inherits a constraint that prevents it from being assigned to the same user as either the Engineer or Engineering Manager role. The converse is also true.

Given that constraints are inherited, any role senior to those shown above could inherit from either the Operations or Engineering Manager roles, but not from both.

Role constraints are a powerful feature of RBAC, but their introduction requires very careful planning, especially in a general hierarchy. Therefore, it is recommended that role designers leverage limited inheritance hierarchies or conditional inheritance wherever constraints are applied, as illustrated in Figure 16 and Figure 17.

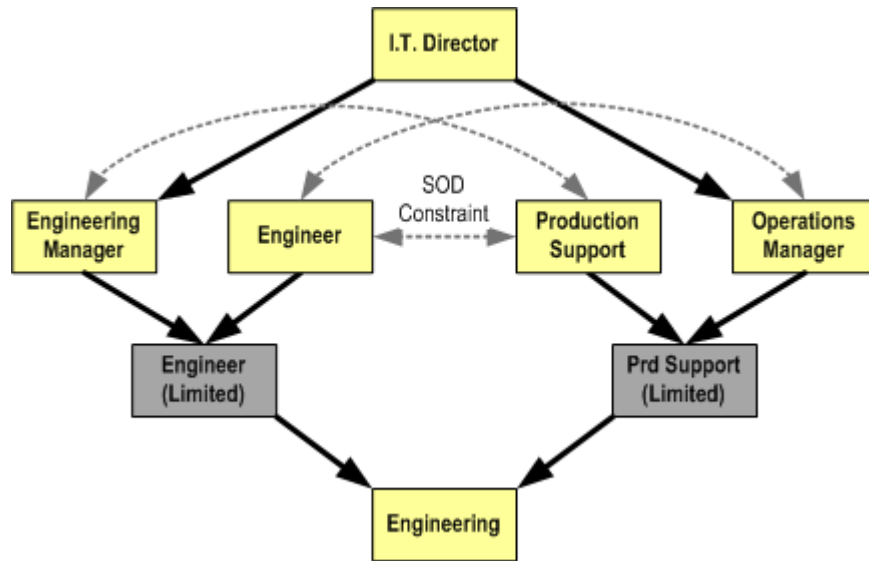


Figure 16: Using Constraints in a Limited Inheritance Hierarchy

By moving sensitive privileges into non-inheritable roles as demonstrated here, constraints can still be applied to junior roles without impacting the ability of more senior roles to aggregate entitlements.

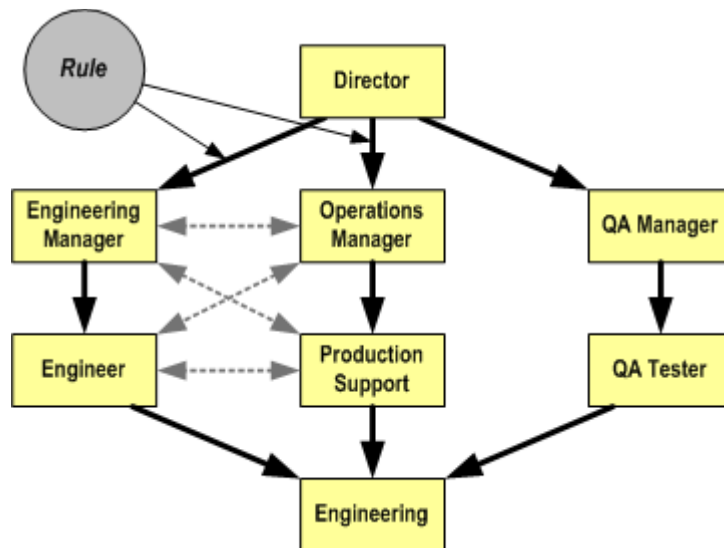


Figure 17: Using Constraints with Conditional Inheritance

The alternative is to use conditional inheritance, if it is available. In this scenario, a rule could be applied that would allow the Director role to inherit from either the Operations Manager or Engineering Manager, but not from both.

3.3.4 Level 4 – Symmetrical RBAC

Provides a permission/role review and auditing interface for role administrators.

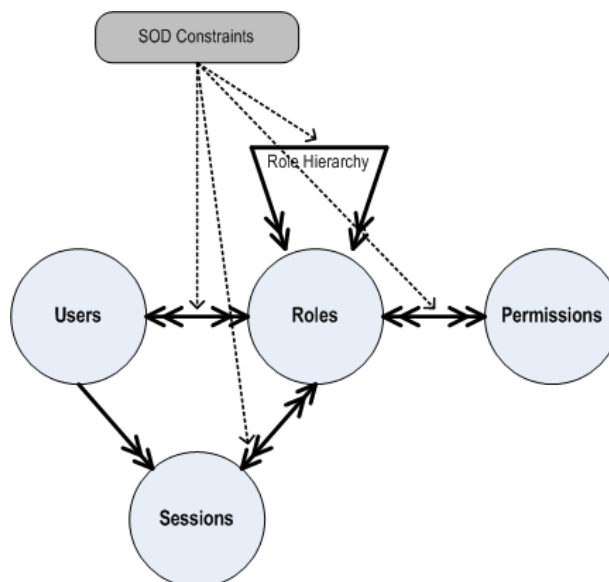


Figure 18: Symmetrical RBAC

Symmetrical RBAC enables the introduction of constraints to permission/role assignments, thus ensuring that no role or user may possess conflicting permissions. It provides for an interface that enables administrators to review permission assignments in the context of a specific user or role. These permissions may be assigned directly or indirectly.

In most modern RBAC-compliant identity management systems, the basic features of symmetrical RBAC are assumed. One of the major use cases for permission assignment reviews is to ensure that permissions are revoked in a timely fashion once they are no longer required, such as when a user leaves the enterprise. If permission assignments are automated by a user provisioning system, either by using roles or by assigning permissions directly to a user, access revocation should be automatic.

This is not to say that permission/role and permission/user reviews aren't an important element of effective governance, as there are always cases where permissions need to be granted independently of an IAM system, and may therefore fly under the radar. Such exceptions will need to be reviewed on a periodic basis.

4. Practical Advantages and Limitations of RBAC

In the real world, role projects often fail to meet stakeholder expectations. This can be due to many factors, such as ownership issues, political bottlenecks, weak governance, inefficient role design, lack of stakeholder participation, inadequate tools and products, failure to engage an experienced implementation partner, poor planning, or unrealistic expectations regarding the ability of RBAC to provide entitlements coverage.

Even though the theory behind RBAC is relatively simple, the complexity of introducing roles to a mature enterprise should not be underestimated. Role projects are labor intensive, highly complex and almost invariably encounter challenges that are difficult to anticipate.

While the advantages of RBAC are self-evident, it is critical to maintain a clear-eyed perspective of its limitations. Ultimately, RBAC is a governance framework, not a software product that can simply be installed. Even the most sophisticated IAM tools are of no use if business processes are poorly defined. Just as roles cannot be created in a vacuum, neither can they survive in one. For long-term sustainability, a robust governance framework is essential, and must include clear process definitions for managing the role lifecycle (conception, creation, modification, attestation, remediation, rationalization and termination).

4.1 Advantages of RBAC

Although a role project is one of the most complex IAM initiatives that an organization can undertake, requiring careful planning and cross-organizational collaboration, there are compelling advantages to a well-conceived RBAC implementation.

- 1) For user populations that share common entitlements, roles can simplify, streamline and automate expensive security administration processes, particularly when implemented in conjunction with a holistic identity management solution.
- 2) Role constraints help to avoid SoD violations. These constraints can be applied in both a preventative and detective fashion, depending on business need and the tools available. Preventative controls preclude assignment of conflicting roles or permissions, while detective controls identify users in possession of conflicting roles or permissions.
- 3) Once implemented, RBAC requires low maintenance and has proven to be extremely scalable in large corporate environments across a broad range of industry verticals.
- 4) In the event that entitlements need to change for an entire group of users, changes can easily be applied to all role members by updating the role definition. Such scenarios include organizational changes, new applications, changes to a role hierarchy and modified job responsibilities.
- 5) Roles provide greater visibility into who has access to what and the ability to ensure that users are granted only the minimum privileges required by their job function. This reduces the operational and reporting burden of satisfying compliance mandates, and provides the business with effective control over access to sensitive information assets.

4.2 Limitations of RBAC

As previously noted, it is important not to view RBAC as a panacea for an organization's access control challenges. Following are some of the limitations that need to be considered at the inception of a role project.

4.2.1 Anticipating Human Factors

In the enterprise, IAM projects invariably encounter organizational and political bottlenecks, and role projects are no different. A major outcome of role engineering is the rationalization and, where necessary, the removal of existing privileges, especially where employees have accrued more permissions than necessary through various job responsibility changes.

Revoking privileges can be threatening to users. Even if they are no longer required, removing them may create uncertainty and fear. A robust communications plan should aim to address any misconceptions.

The introduction of roles often requires reengineering of business processes, and is therefore disruptive to those currently engaged in such processes. This is particularly true of security administrators, who often feel a sense of proprietorship over legacy processes and may be reluctant to surrender their authority. To a lesser extent, the same is true of system and data owners, who may express reservations about an IAM product connecting to their resources and managing permissions within it.

Such bottlenecks are difficult to overcome unless the project team is empowered by an executive mandate. For enterprise IAM projects such as RBAC, strong executive backing should be considered obligatory due to the political and organizational resistance that the project is likely to encounter.

4.2.2 Planning for Role Exceptions

It is almost impossible to find a context in which roles can accurately describe 100% of an organization's entitlements, as some exceptions are unavoidable. Consider, for example, a system administrator who may require temporary access to a server for troubleshooting purposes. It is clearly impractical to define unique roles for every such exception. Doing so would eventually result in an unmanageable number of roles. Thus even organizations with mature RBAC implementations find that some discretionary access controls are still required.

Exceptions are also common among managers and executives, who by definition tend to have privilege needs that are too unique to warrant the creation of a dedicated role. A director, for instance, may acquire some privilege needs by virtue of a role inheritance hierarchy, but would also likely require additional privileges that are not inherited, such as access to certain HR records or management reports. While it is possible to create roles that cover such exceptions, the value of doing so is questionable.

Of course, if exception-based DAC processes are still permitted, then the opportunity for policy violations remains, albeit to a much lower extent. This is why a GRC tool is an important component of a holistic IAM solution, as it will enforce policies in a detective fashion and enable closed-loop remediation of policy violations.

4.2.3 Preventing Role Explosion

Roles tend to deliver greater value where a large subset of the user population shares common privilege needs. Such examples include retail sales clerks, bank tellers, call center operators and helpdesk technicians. The ability of roles to provide for a user's privilege needs is inversely proportional to how specialized their job function is.

A common reason for the failure of many RBAC implementations is that inexperienced role designers attempt to create unique roles for every conceivable exception. This leads to what is commonly known as "role explosion", where an RBAC implementation can lead to the creation of more roles than users. Role explosion is a recipe for disaster, as the number of roles quickly becomes unsustainable. Role engineers should therefore be careful to avoid such diversions and focus instead on roles that provide the greatest coverage and thus deliver the greatest business value.

Preventing role explosion in the design phase of an RBAC implementation requires clearly defined role standards and policies. At a minimum, these standards should include:

- a) The minimum number of users sharing common privilege needs to warrant creation of a role. This metric is usually refined as the role design matures, but the number should be low enough to cover a reasonable percentage of the organization's entitlements and high enough to prevent role explosion.
- b) The target percentage of an organization's entitlements that should be covered by roles. In most organizations, 80% is a reasonable target. The 80/20 principle is discussed in section 4.2.3.1.
- c) Standard role naming conventions to prevent ambiguity and duplication, and also to provide contextual relevance.

A robust identity management suite will offer additional features that can help to prevent role explosion, such as support for attribute-driven permission/role assignments, attribute aware roles, contextual permission/role assignments and session-data based user/role assignments. Attribute aware roles, as discussed in section 3.2.2, are particularly important in helping to prevent role explosion.

On an ongoing basis, strict role governance is essential. This should incorporate a role review process to ensure that:

- a) Existing roles are recertified on a periodic basis
- b) Exception-based roles are not created arbitrarily
- c) Obsolete roles are deleted from the role catalog on a periodic basis
- d) Duplicate roles are not created for an entitlements set that is already represented
- e) New roles are descriptive and have business relevance
- f) All roles have assigned owners

4.2.3.1 The 80/20 Principle

Given the exceptions that invariably exist in every organization, it is critical to maintain a sense of realism about the ability of RBAC to provide for user privilege needs. The 80/20 principle is based on the guiding principle that a well-conceived RBAC implementation should cover approximately 80% of entitlements within most organizations, with exceptions accounting for the remaining 20%. Role explosion usually occurs when weak role governance permits the creation of unique roles to support such exceptions; the difference between 80% and 90% coverage of an organization's entitlements can increase the overall number of roles by an order of magnitude.

The 80/20 principle assumes wide variation between different user populations. Managers, executives and other specialized employees may find that roles accurately describe just 30-40% of their privilege needs, but this is typically offset by large numbers of lower level staff who share common entitlements. Among such user populations, roles may describe 90% or more of privilege needs.

Of course, policy violations can still occur even where a mature RBAC framework has been implemented. Discretionary access can still be provided on

an exceptional basis, or by using “backdoor” native provisioning mechanisms such as a directory administration console.

A well-conceived RBAC implementation therefore requires GRC tools that can flag policy violations in both a detective and a preventative fashion. The topic of tool selection will be addressed later in this document.

4.3 Is ABAC an Alternative to RBAC?

While RBAC focuses on bundling entitlements into descriptive roles, ABAC (Attribute Based Access Control) provides contextual authorization of users based on permission-granting attributes. Permissions are granted based on assertions made by a user when access is being requested to a system. An assertion at the very least describes who the user is, and what operation they are attempting to perform, but can also contain any attributes about the user pertinent to the request being made. ABAC is a declarative form of access control, as opposed to RBAC, which is descriptive.

XACML (Extensible Access Control Markup Language) is an OASIS standard for a policy control language that implements ABAC. It provides an XML schema for resources that support ABAC-compliant authorization.

There is a legitimate school of thought in the IAM community that ABAC is in fact an alternative to RBAC, or even an evolution of it. We disagree with this notion, since the distinguishing features of ABAC now have functional equivalents in role products that support attribute-aware, dynamically assigned and contextually-aware roles. Furthermore, there is an enormous operational overhead involved when managing ABAC in a large, heterogeneous environment without an overlaying descriptive framework such as RBAC to provide business relevance.

However, that is not to say that ABAC, and XACML in particular, should be disregarded. RBAC, as previously noted, is a descriptive form of access control and therefore does not provide support for the actual authorization of a user based on policy enforcement. Therefore, we suggest that in some scenarios, ABAC compliments and enhances RBAC, particularly when there is a need to describe fine-grained permissions and perform dynamic, contextual authorizations.

The full scope of ABAC/XACML, and the practical implications of integrating and managing it within an RBAC framework, is an extremely complex topic that is far beyond the scope of this paper.

5. Role Modeling

Role modeling is the process of creating descriptive roles that accurately describe the privilege needs of a target population. There are three widely accepted approaches to role modeling:

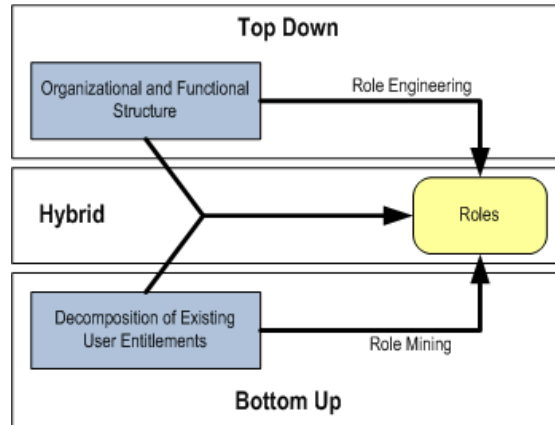


Figure 19: Top Down, Bottom Up and Hybrid Modeling Techniques

5.1 Top-Down (Role Engineering)

Role engineering discovers roles that are influenced by business and organizational factors, and thus tend to align with job titles and organizational hierarchy. Accordingly, such definitions are often termed “functional” or “business” roles. In a top-down approach, business stakeholders are the major influencers of role design. HR data is also a common source for organizational and functional roles, and for identifying role candidates. The concept of business roles was discussed earlier in section 3.1.1.

Given that RBAC is based on the principle of least privileged access, top-down modeling tends to result in the creation of extremely coarse grained roles that may have relevance and meaning to the business, but which describe a relatively small percentage of user privilege needs.

While top-down role engineering is valuable in establishing high-level organizational hierarchies, it is a highly manual and time consuming exercise that requires extensive participation from business stakeholders, particularly at the functional level. Without a decomposition of existing entitlements, top-down can result in vast discrepancies between the entitlements that stakeholders think users *should* have and the entitlements that users *actually* have. The ROI for role projects that depend entirely upon a top-down approach, without any rationalization of existing entitlements, tends to be extremely low.

5.2 Bottom-Up (Role Mining)

Role mining has emerged as a popular technique for discovering roles and role candidates. This approach involves the use of an analytics tool to decompose existing user entitlements and create roles based on common privileges among a target population. Many commercial role mining tools provide the ability to import entitlements from a flat file, a data warehouse or even from an existing user provisioning system. Such tools usually allow thresholds to be applied to raw entitlements data, enabling the administrator to model different outcomes.

Role mining is an effective method for discovering fine-grained technical roles for narrowly defined user populations. Once role candidates are identified, their identity attributes can be evaluated for common denominators; these can be used to create automation rules for role assignments.

The major shortcoming of role mining is that the discovery process relies upon the quality of existing entitlements data. If the quality of entitlements data is inadequate, then the discovered role candidates are also likely to be incorrect, or at least inconsistent.

By definition, any enterprise considering RBAC is likely to have an existing entitlements problem, so data quality is probably low. In organizations that rely on DAC processes, user entitlements are typically cumulative, which adds to the unreliability of source data.

Role mining is therefore best suited to narrowly defined population sets within individual systems and applications. If a role project is entirely reliant upon analytics for role discovery, then it must incorporate a preparatory data cleansing phase to ensure that the generated roles adequately describe the privilege needs of users.

5.3 Hybrid

Due to the limitations of relying upon top-down or bottom-up modeling techniques, a hybrid approach is recommended. This approach combines role engineering and role mining to produce more relevant and accurate outcomes.

In a hybrid model, coarse-grained business roles are defined to identify target populations in a business relevant manner, typically in alignment with an organizational structure. These roles are then rationalized and refined by mining target systems for common entitlements. This approach allows business roles to more accurately reflect the privilege needs of users by incorporating finer grained technical roles, and is even more likely to yield accurate results when examining limited subsets of users at the functional level.

Another major benefit of a hybrid approach is that it facilitates the phased introduction of roles; one population at a time, and one system at a time.

6. Planning a Roles Project

Careful planning is the most important activity of any RBAC implementation. Many RBAC projects are doomed to failure from the inception stage due to unrealistic expectations, lack of stakeholder participation, flawed modeling techniques or poor governance.

Even with the best tools available, an enterprise role project is extremely labor intensive, and the level of effort typically involved in RBAC implementations should not be underestimated.

At a high level, an enterprise role project should include the following phases:



6.1 Establish the Business Case

A well-crafted business case is critical for obtaining executive sponsorship and funding. Executive sponsorship is necessary to drive stakeholder participation and avoid organizational bottlenecks. IAM projects are notorious for being politically contentious, since they invariably impact ownership of legacy systems, data and processes. Additionally, the concept of process automation can be threatening to individuals who are currently performing manual identity and access administration tasks. Without robust executive sponsorship to mandate an RBAC strategy, it is difficult to obtain the cooperation of process owners and maintain stakeholder participation.

The business case for RBAC is typically quantified in the form of an ROI, incorporating the following metrics:

Benefit	Measure	Cost Metrics	Factors
Reduction in operational overhead	Reduced time taken assigning permissions to new hires	Hourly operational labor costs (fully loaded)	Number of new hires per year
	Reduced time taken updating permissions for job responsibility changes	Hourly operational labor costs (fully loaded)	Number of job changes and transfers per year
	Reduced time taken assigning permissions to existing users	Hourly operational labor costs (fully loaded)	Number of access requests per year
	Reduced time taken to revoke user access	Hourly operational labor costs (fully loaded)	Number of terminations per year
	Reduced labor costs from automation of compliance reporting	Average hourly labor costs (fully loaded)	Number of compliance reports per year
	Reduced labor costs from more streamlined access recertification processes	Average hourly labor costs (fully loaded)	Number of recertifications per year
Improved Productivity	Improved productivity for new hires waiting for access to be provisioned	Productivity gained (%) multiplied by average hourly labor costs (fully loaded)	Number of new hires per year
Risk Reduction	Reduction of security exposures and audit/compliance violations	Likely cost to the company of a regulatory fine or security breach	Number of possible violations identified per year

To avoid setting unrealistic expectations, it is always advisable to adopt a conservative approach when calculating ROI. The ROI metrics be used to measure the project’s success or failure. Project leaders should speak with industry professionals, integration partners and even other companies that have successfully implemented RBAC to support any cost/benefit assumptions that are included in the business case.

IAM projects that are launched without strong executive sponsorship frequently encounter insurmountable political and organizational challenges, so a compelling business case is essential to gaining senior management backing. An experienced implementation partner can assist in building the case and positioning the project for success.

6.2 RBAC Readiness Assessment

The complexity and duration of an RBAC project depends, among other things, upon the maturity of an organization's existing IAM processes and tools, the size of the user population, the capabilities of the internal project team, and the number of systems and entitlements in the target environment. The following factors should be considered when performing an RBAC readiness assessment:

1. The presence of existing identity management tools and technologies.
 - a. How many target resources will be actively managed by roles?
 - b. Do existing tools support all four levels of RBAC service?
 - c. Do existing tools support extended RBAC features such as security aware roles, role inheritance, conditional inheritance and role attributes?
 - d. Do existing tools incorporate a rules engine that can be used to automate user/role assignments?
2. The quality and availability of HR data and other authoritative sources of identity data.
3. The cleanliness of existing entitlements data in target systems.
4. The number of individual entitlements in the organization.
5. The number of roles already defined at the enterprise, functional or sub-system level.
6. The number and accuracy of existing access control policies
7. The capabilities and experience of the internal role project team, and their ability to be fully dedicated to the project.
8. The commitment of key business and I.T. stakeholders to participate.

Most organizations tend to be subjective when performing self-assessments, as they very often lack a sufficient frame of reference against which to benchmark their own maturity. Once again, an experienced implementation partner can provide an independent, objective and candid assessment based on their knowledge of industry best practices and by benchmarking the organization against others who have engaged in RBAC implementations.

6.3 Select Tools and Implementation Partners

Like most IAM disciplines, RBAC has undergone significant maturation over the past several years, both in terms of products and best practices. Nevertheless, choosing the wrong tool or implementation partner can prove to be an expensive mistake.

6.3.1 RBAC Tools

There are numerous factors to consider when selecting tools for a role project. Although the IAM product space has matured in recent years, it is nevertheless daunting for the uninitiated. Most vendor offerings require a steep learning curve for internal staff, and some have combined user provisioning, compliance and role management features into a unified suite.

It is not the purpose of this document to recommend specific products or vendors, as each organization's needs, internal capabilities and budgetary constraints are different. But at the very least, a well-conceived RBAC implementation should implement the following components:

6.3.1.1 User Provisioning

Enables the automated provisioning and de-provisioning of user accounts. It should include the requisite connectors/adapters for target systems and support all four levels of role service. Depending on the project requirements, it should also support features such as basic role inheritance, conditional role inheritance, SoD policy enforcement, role attributes, attribute aware roles and rule-driven user/role assignments.

The user provisioning system is critical to the automation of user/role and permission/role assignments, and selecting a tool with a limited feature set can result in additional project complexity and cost if the project team is required to work around its limitations.

Examples of User Provisioning Systems:

- Oracle Identity Manager
- Oracle Waveset (formerly Sun Identity Manager, now deprecated)
- Novell Identity Manager
- CA Identity Manager
- Tivoli Identity Manager
- Sailpoint IdentityIQ

6.3.1.2 Role Mining and Analytics

Provides an analytics capability to examine user entitlements within existing systems and model different outcomes based on thresholds defined by the administrator. Also allows role designers to perform impact analysis when creating new roles, and provides a user interface for periodic role reviews.

Role mining should not be used exclusively when discovering roles, but it is nevertheless an important element of the role modeling process.

Examples of Role Analytics Tools:

- Oracle Identity Analytics (formerly Sun Role Manager)
- CA Role and Compliance Manager
- Sailpoint IdentityIQ
- Aveksa Role Manager

6.3.1.3 Governance, Risk and Compliance

Performs periodic and on-demand reviews of user/role, permission/role and user/permission assignments to detect policy violations. The GRC tool should have the ability to generate compliance reports, in addition to automating closed-loop attestation and remediation processes. Ideally, it should also provide native integration with the user provisioning system to perform real-time risk and impact analysis when a provisioning activity is taking place.

Examples of GRC Tools:

- Oracle Identity Analytics
- Aveksa Compliance Manager
- CA Role and Compliance Manager

- Sailpoint IdentityIQ

6.3.2 Implementation Partners

Although it is possible to deliver an enterprise RBAC solution without external assistance, the risk and complexity inherent to this kind of project make it extremely unwise to do so. An experienced implementation partner can leverage their experience of prior RBAC implementations to help an organization avoid pitfalls, navigate logistical and organizational challenges, devise an effective implementation strategy that delivers incremental benefits, and ensure that the design complies with industry best practices.

6.4 Baseline Role Analytics

With the assistance of an implementation partner, the next step should involve top-down modeling of coarse grained business/organizational roles. A good place to start is with base enterprise roles such as Employee and Contractor, and then traverse down the organizational hierarchy to construct a high-level inheritance hierarchy.

Every organization is different, thus discretion should be used when deciding how deep to go when defining organizational roles. Most organizations are subject to periodic restructuring; as a general rule, departments are more likely to be impacted by reorganizations than business units and divisions. Thus, finer grained business roles are more prone to change.

A role analytics tool is used to identify existing entitlements that can be aggregated into roles, and will assist in the ongoing cleansing of privilege data. In the early stages of a role project, analytics should be performed against individual systems at the organizational level rather than at the functional level. This approach is most likely to result in quick wins, where the introduction of high-level organizational roles can deliver some rapid benefits. Attempting to discover roles that cross organizational or technology boundaries is unlikely to yield significant benefits prior to completion of a data cleansing effort.

Ultimately, the main objective of a baseline analytics exercise is to gauge the level of effort that will be required to implement an enterprise role framework. While some quick wins should be gained from data cleansing and the creation of organizational roles, this phase should still be considered a design and requirements gathering exercise.

An experienced business analyst should be engaged to lead the analytics effort and document requirements. Generally, role modeling and requirements gathering efforts are less successful when performed by technical staff, who may lack the training and experience to ask the right questions of business stakeholders. By definition, enterprise role projects should be more concerned with business need than technology considerations.

6.5 Identify Project Participants and Stakeholders

There are typically three major components to consider when identifying project participants and stakeholders. First is the project team itself, which will be responsible for implementing the solution. Second is the governance committee, which will be responsible for policy and oversight. Third is the stakeholder committee, which will include participants from I.T. and the business.

6.5.1 Core Project Team

Although job titles and functions will vary, the following competencies are obligatory for an RBAC project:

6.5.1.1 Project Manager

Oversees the core project team. Responsible for project planning, communication strategy, escalating risks and issues, maintaining ROI, management reporting, communication strategy and engaging appropriate stakeholders.

6.5.1.2 Solution Architect

Provides technical oversight and leadership for the implementation team, ensuring compliance with best practices. Responsible for producing technical blueprints and build/design specifications. Acts as an escalation point for technical issues and has ultimate accountability for the solution.

6.5.1.3 Implementation Engineer(s)

Responsible for implementing the solution in accordance with defined business requirements and technical blueprints. Configures and customizes any products required to support the solution.

6.5.1.4 Business Analyst

An individual with outstanding written and verbal communication skills and who can demonstrate strong analytical capabilities. Responsible for requirements gathering, role modeling, and business process reengineering in collaboration with identified business and technical stakeholders. Documents and maintains requirements, business processes, role definitions, relationships and components.

A new function of Role Owner will also need to be created for each role:

6.5.1.5 Role Owner

May be either a business or I.T. person, depending on whether the role is a business or technical role. The role owner should be an individual who has a deep understanding of the entitlements represented by a role. Responsibilities will include discretionary approvals of new members, performing periodic role reviews/recertifications/attestations, and initiating requests to the Role Governance Committee for role creation and modification.

6.5.2 Governance Committee

Robust governance is essential to the long-term sustainability of an RBAC implementation. Without clearly defined policies and standards, supported by a strict enforcement apparatus, role initiatives can quickly degrade, even if the initial implementation is successful.

The governance committee is typically a mixture of business and I.T. stakeholders, and by definition must continue to operate after completion of the project. During the implementation phase, they will have authority over project scope and direction, but can also be invaluable in ensuring that the project gains visibility across the organization.

As the role framework evolves, the committee will be responsible for definition and enforcement of policies and standards, maintaining the enterprise role catalog, assigning role owners, approving role creation/change/termination requests, and ensuring that new systems and applications are adequately represented by the RBAC framework.

Typical participants might include senior leaders from I.T. Security, Operations, Corporate Audit, Risk Control, HR and each major business unit. Role owners should have a matrixed reporting relationship to the Governance Committee, but are typically not full members themselves.

6.5.3 Stakeholder Committee

The stakeholder committee should comprise I.T. and business participants who are responsible for systems, business processes and tools that are likely to be impacted by a role project. The main purpose of the committee is to facilitate planning, communication and change management as business processes are re-engineered to enable the introduction of roles.

Usually, the stakeholder committee will be larger than the governance committee, since not every business unit will be large enough to warrant representation on the governance committee.

6.6 Articulate Goals and Objectives for the Project

As with any enterprise project, it is important to understand why an RBAC implementation is necessary, what specific business challenges it aims to address, and to define success metrics that are both measurable and achievable. The data gathered from the Readiness Assessment and the Baseline Analytics exercise will inform the definition of realistic goals and objectives.

Business and technical stakeholders must agree on these goals and objectives, which at the very least should include the following:

6.6.1 Implementation Approach and Strategy

A well-defined strategy should include a pilot phase and assume a phased implementation. Phasing the introduction of roles helps to demonstrate ROI, builds project momentum, mitigates risk, maintains stakeholder engagement and delivers incremental business value.

6.6.2 Measurable and Clearly Defined Success Metrics

Using the data gathered from the Readiness and Analytics phases, the metrics defined in the business case should be refined accordingly. These are the criteria by which the success of the project will be measured, thus a conservative approach is recommended to avoid setting unrealistic expectations. Other than obvious ROI factors, success metrics may include the target number of systems and entitlements that will be covered by roles, the target number of roles and the number of business processes that will be streamlined or fully automated.

6.6.3 Project Plan and Timeline

An enterprise role project can take anything from a few months to 2-3 years to complete, depending on the scope, ambition and complexity of the effort. In general, the average timeline for most role projects is 6-12 months assuming that best practices are observed.

The project plan should reflect the availability of key stakeholders and participants, who may not be fully dedicated to the project. It should also represent a phased approach, so that business stakeholders can plan appropriately for the introduction of roles to their respective departments.

6.6.4 Communication Strategy

A well-defined communication plan is important for ensuring stakeholder participation, raising general awareness about the role project, and for preempting any misconceptions about what the introduction of roles means to individual users. The latter consideration is particularly important, as RBAC can generate uncertainty and resistance among end users, security administrators and data/process owners. If these factors aren't preempted early in the project cycle, problems can arise later in the project lifecycle.

The communication plan should also include a clear definition of what roles are, the advantages of RBAC, and provide a forum for encouraging user feedback and suggestions.

An effective communication strategy is to embark upon a "road show", comprising RBAC forums for user groups throughout the organization. This is an effective strategy for addressing misconceptions, alleviating fears, maintaining visibility, and also for gathering feedback from end users, who are often a source of valuable insights and can identify use cases that might otherwise be overlooked.

6.7 Iterative Implementation

As noted in 6.6.1, a phased implementation strategy is strongly recommended for role projects. This helps to demonstrate ROI, maintain project momentum and visibility, mitigate risk and manage scope. So called "big bang" implementations are rarely successful, as it is typically so long until a solution is actually implemented, the organization may have gone through major changes since project inception, requiring modifications to project assumptions, scope and requirements.

A pilot phase is strongly recommended. Start with a relatively uniform population where the probability of discovering common entitlements is high. Good candidates for a pilot phase include bank tellers, helpdesk technicians, call center operators and sales assistants. The pilot phase will deliver a quick win for the project and should be used to help refine the processes and standards for managing the role lifecycle.

Iterations should focus on individual business units and departments, and then individual systems and applications, using a hybrid approach to role modeling. Top-down role engineering can be used to create coarse-grained functional roles and identify role candidates, while bottom-up role mining can be used to discover technical roles within a target population and rationalize entitlements. The concept of hybrid role modeling, which combines both approaches, is discussed in section 0.

In parallel with an iterative implementation, the project team will also need to consider use cases such as, for instance, how to handle transfers during a job transition, how to prevent access being revoked when a contractor becomes a full-time employee, or how to restore a former employee's previous access after an extended leave of absence. Such use cases are typically enterprise-wide and should therefore be identified and implemented as part of a baseline RBAC framework.

In a typical implementation, each iterative release should comprise the following phases:

6.7.1 Identify the Target Population

The target population may be defined by a business unit, department, job function or even a physical location. HR data is a common source for identifying role candidates and job functions. Stakeholders will be able to validate assumptions about organizational and functional relationships to ensure that role candidates are accurately represented.

6.7.2 Identify and Prioritize Target Systems

I.T. and business stakeholders can usually provide valuable insight into what systems and applications incur the highest operational burden. Focus on the areas of greatest business value when prioritizing. Enterprise directories, databases and audit-relevant systems such as ERP are usually good "quick win" candidates.

6.7.3 Data Validation and Cleansing

Mining existing entitlements for the target population within each system may reveal anomalies such as missing or orphaned privileges that can be cleaned up. This will improve the accuracy of identifying role candidates. Role analytics may also help to identify false positives (users who have a privilege that shouldn't) and false negatives (users who should have a privilege but don't).

6.7.4 Role Modeling

Once data cleansing has been completed, bottom-up mining of each system will help to identify candidates for technical roles. These can then be nested into the appropriate business roles. Role modeling requires consideration of SoD policies, exceptions, and security relevant attributes in target systems. Owners should be identified for each identified role and provided with the relevant product training.

Role designers should aim to minimize the number of discovered roles through aggregation and sharing. As more systems are analyzed for each population, opportunities will be identified for further aggregation of technical roles.

6.7.5 Rule Creation

Identity attributes belonging to members of each discovered role should be analyzed to generate rule-driven role assignments where possible, while ensuring that rules do not incorrectly assign roles to users. Rules should always be reviewed and validated by the appropriate business or technical stakeholders.

6.7.6 Implementation

Creation and integration of each identified role using the IAM tools available. Roles and service levels should be defined within the role catalog, while business processes and membership rules need to be clearly documented.

6.7.7 Clean-Up

Anomalies or exceptions, such as minor, non-critical systems where roles are not easily discoverable, should be identified and documented accordingly. The target population should be reexamined for entitlements that have not been covered by roles, and these entitlements should either be removed or documented as exceptions if still required. The clean-up phase can also be used to eliminate superfluous technical roles by aggregating them where appropriate.

6.7.8 Completion

At the end of each release, metrics should be collected on the percentage of entitlements covered by roles, the level of automation achieved, issues encountered and lessons learned. The role catalog should be updated, and metrics provided to the governance committee.

7. Conclusions

In this paper, we have provided a high-level overview of RBAC, expanded on the ANSI/INCITS 359-2004 standard, and described some best practices and recommendations for introducing roles to the enterprise. It should be noted that RBAC is merely one possible approach for enforcing access controls, and there is still legitimate debate about whether it is effective in every scenario. However, the increasing maturity of RBAC tools and best practices has addressed many of the issues that once plagued many role projects. The successful implementation of RBAC in major financial institutions, healthcare providers and federal government agencies such as the Department of Defense and the Department of Veterans Affairs, serves as testament to the scalability of RBAC in complex, heterogeneous environments.

Interest in RBAC is driven largely by cost considerations, operational inefficiencies and a desire to reduce organizational risk. Increasingly stringent regulatory mandates, the growing complexity of information systems and the ever-present threat of insider attacks—which constitute 70% of all known breaches—have forced organizations to confront their entitlements management challenges. RBAC offers a compelling set of capabilities that can streamline security administration, reduce organizational risk, satisfy compliance mandates, address operational inefficiencies and deliver significant ROI. Indeed, an independent study by the Research Triangle Institute revealed that RBAC saved U.S. industry \$1.8 billion in 2009 alone.

Nevertheless, some misconceptions about RBAC remain, not least of which is the expectation that roles can be implemented simply by purchasing the right tool. While there are a number of highly sophisticated commercial products that facilitate role lifecycle management and simplify role discovery and implementation, none of these should be considered a “silver bullet”. Tools are no substitute for well-defined business processes, strong executive sponsorship and a robust governance framework. Without these three ingredients, role projects are unlikely to succeed.

When RBAC is planned and implemented correctly, it provides an extremely scalable and low maintenance approach to entitlements management and has proven effective in medium to large organizations across a broad range of industry verticals. More importantly, it provides business stakeholders with a sense of accountability and control over who has access to sensitive information assets, and equips them with the tools to ensure that privileges are granted only to those who need them.

In conclusion, companies who are seeking to streamline security administration processes and reduce organizational risk should consider RBAC an integral element of any holistic IAM solution.

About the Author

Toby Emden leads the Identity Management practice for Qubera Solutions. He has more than ten years experience as an I.T. Security professional in various technical leadership and governance roles, focusing on Identity and Access Management, Role Engineering, Cryptography, Network Security, Data Privacy and Regulatory Compliance. Prior to joining Qubera, he was Chief Security Architect and VP of the Security portfolio for a Fortune 100 financial services company.

Bibliography

1. **Sandhu, Ravi S., et al.** *Role-Based Access Control Models*. George Mason University and SETA Corporation. s.l. : IEEE, 1996.
2. **Hilchenbach, Burkhard.** *Observations on the Real-World Implementation of Role-Based Access Controls*. s.l. : Schumann Security Software, Inc.
3. **Novell Software.** *Implementing Continuous Roles-Based Access Governance*. 2009.
4. **Bowron, Ron.** *Role-Based Access Control*. s.l. : Big Sky Associates, 2009.
5. **Vanamali, Srinivasan.** *Role Engineering: The Cornerstone of Role-Based Access Control*. s.l. : CA Services, 2009.
6. **Fuchs, Ludwig and Pernul, Gunther.** *HyDRo - Hybrid Development of Roles*. University of Regensburg. 2008.
7. **ANSI/INCITS.** *359-2004 Standard for Role Based Access Controls*. 2004.
8. **Neumann, Gustaf and Strembeck, Mark.** *A Scenario-Driven Role Engineering Process for Functional RBAC Roles*. Department of Information Systems, New Media Lab, Vienna University of Economics and BA, Austria. 2002.
9. **Ekkebus, Brian.** *Implementing RBAC: Oversight or Out-of-Sight*. *CIO Magazine*. 10/21/2009.
10. **O'Connor, Alan C. and Loomis, Ross J.** *Economic Analysis of Role-Based Access Control*. National Institute of Standards and Technology. s.l. : RTI International, 2010.
11. **Jansen, W.A.** *Inheritance Properties of Role Hierarchies*. s.l. : National Institute of Standards and Technology, 1999.
12. **Kuhn, D. Richard.** *Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems*. s.l. : National Institute of Standards and Technology, 1997.