empowerID

Best Practices in Enterprise Authorization:

The RBAC/ABAC Hybrid Approach

Table of Contents

Introduction to RBAC and ABAC

The modern IT organization is a complex mesh of internally managed and externally hosted applications. A central concern that this situation creates is how best to secure access and control authorization for these applications. A long standing debate in the IT community has been whether *Role-Based Access Control (RBAC)*—granting access to roles—or *Attribute-Based Access Control (ABAC)*—granting access via attribute-based rules—is a better model for authorization management. The one thing that everyone can agree on is that, whatever the model, authorization logic should be created and maintained external to the application and not managed uniquely within each application.

In this white paper we discuss the RBAC versus ABAC models for authorization, lay out the benefits and weaknesses of each approach and offer a hybrid model that retains the benefits of each while avoiding their major weaknesses.

Role-Based Access Control or "RBAC" is a security and authorization model for securing access to computer resources. RBAC is used so widely that almost all large enterprises use RBAC to secure their systems. Sometimes referred to as "Role-based Security," RBAC access is based on roles as defined by the enterprise using them. In the RBAC model, roles are defined and permission sets for resources are assigned to roles. Users are then granted one or more roles in order to receive access to resources.

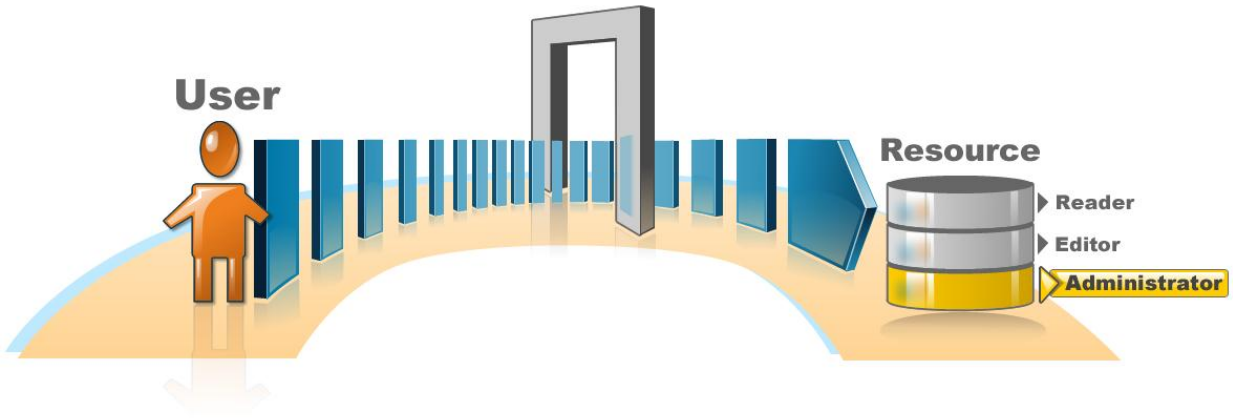

Figure 1. Role-Based access to permissions for resources

ABAC, on the other hand, stands for Attribute-Based Access Control. As suggested by the name, ABAC access control relies on user attributes for access decisions. ABAC policies are rules that evaluate access based upon three sets of attributes. These include: *Subject Attributes*, which are attributes concerning the person or actor being evaluated; *Resource Attributes*, which are attributes of the target or object being affected; and, *Environment Attributes,* which include attributes such as the time of the day, IP subnet, and others that do not relate to either the Subject or the Resource.

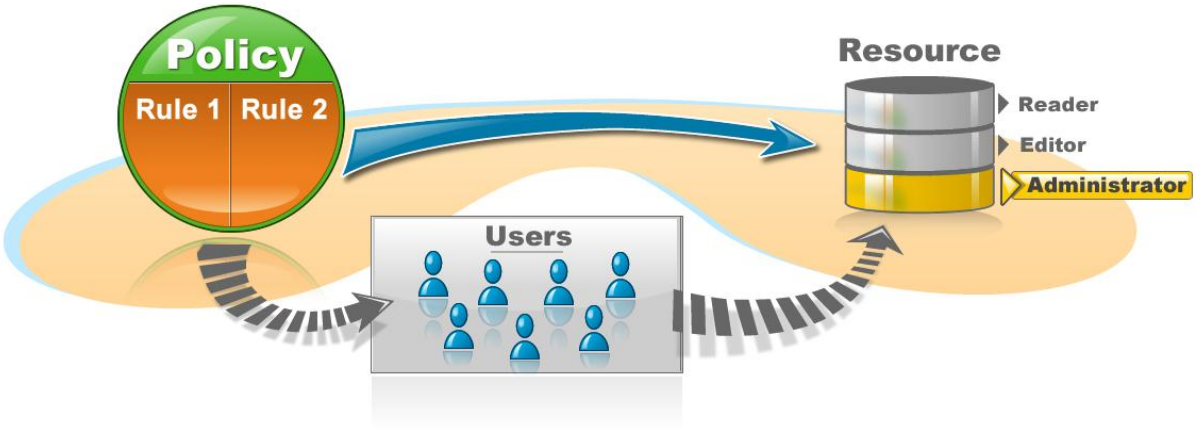

Figure 2. Simple logical view of attribute-based access to permissions for resources

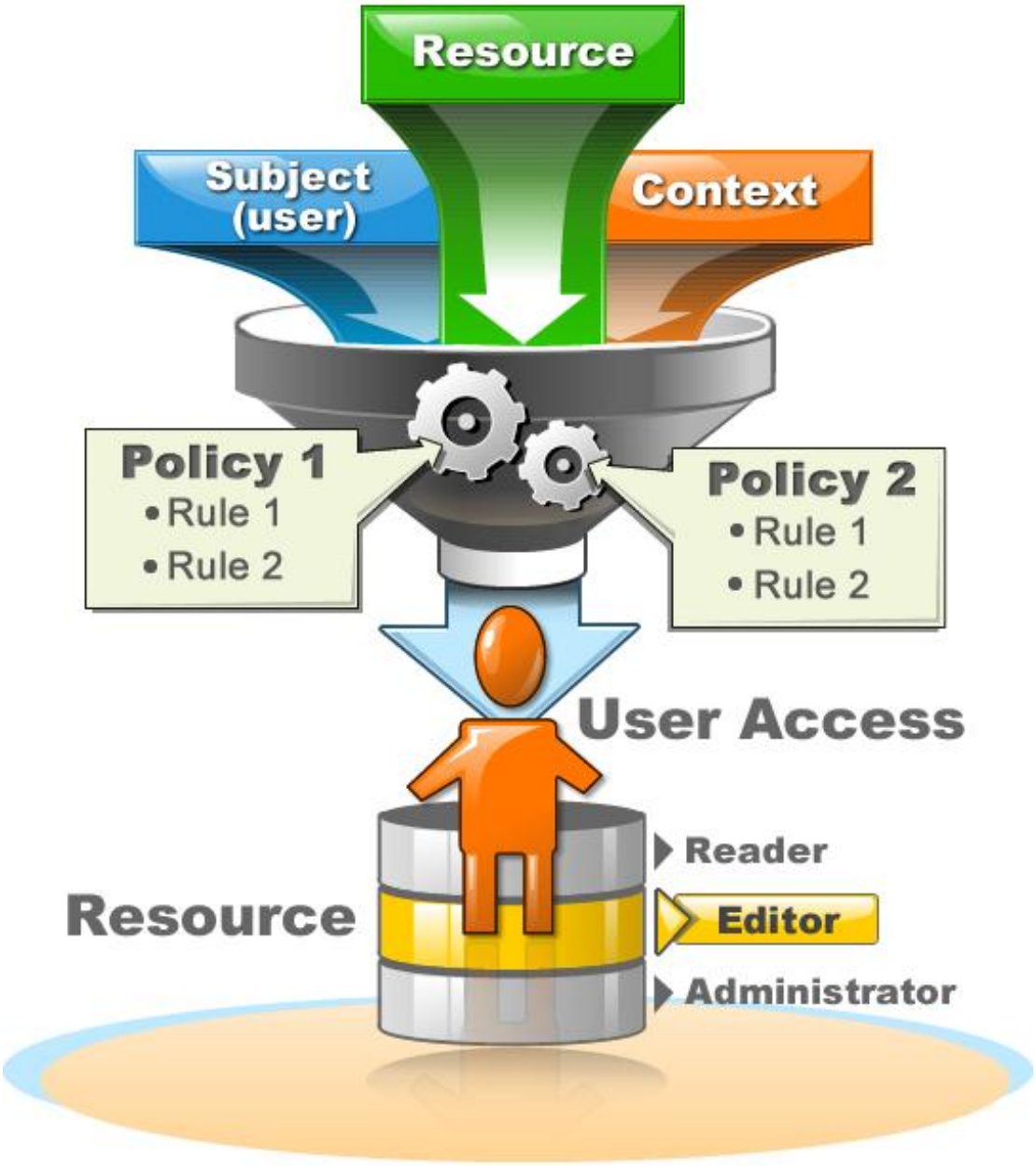

Figure 3. More descriptive funnel-view of how ABAC functions

RBAC Versus ABAC

Generally speaking, RBAC and ABAC each have their own weaknesses and benefits. RBAC trades-off the initial effort of structuring roles for advantages in administration and user permission whereas ABAC reverses those, providing for easier set-up and structuring, but complicating the ease of user permissions review associated with RBAC.

Benefits of RBAC

In the RBAC model, the assumption is that controlling and maintaining access is easier since access is not directly assigned to users but bundled into assignments made to roles. These roles decrease the cost of security management and compliance auditing as they centralize access management into fewer assignments to be managed and audited. In an RBAC model, it is clear who is assigned to a role and what access the role grants. Furthermore, according to a 2010 NIST study, RBAC implementation delivers significant ROI to companies through efficient provisioning and reduced employee downtime

RBAC Primary Benefits:

1. RBAC is deterministic. An RBAC approach makes it easy to know who has access to what at any moment in time.
2. RBAC is more direct and easier to visualize. Security admins are able to visualize the actors and resources that they will affect when creating or modifying a policy.
3. RBAC is inherently auditable. With RBAC assignments it is simple for business owners to certify or attest to access granted as the consequences of that access can be viewed. This contrasts with ABAC where the consequences of a rule are not easy to fully grasp.
4. RBAC can be simpler than ABAC. For example, with RBAC bundles of access can be directly assigned to a user. To do this in ABAC requires the creation of a new rule.

Weaknesses of RBAC

The concrete nature of the RBAC model can also be considered the source of its weaknesses as it is more static than ABAC and doesn't consider contextual information when making an access decision. In order to capture context for access decisions, more roles must be created and a larger amount of data analyzed and preprocessed. For example, if access permissions are differentiated by time of day some users will only have access during normal business hours whereas other users will have access 24/7. Permissions differentiated in this manner could lead to the creation of two different roles: One for normal business hours and one for all-hours access. If other context-related conditions (e.g. authentication strength, remote login, line encryption) were added it could result in the creation of an excessive number of roles (a "role explosion.")

RBAC Primary Weaknesses

1. RBAC typically requires advance knowledge of the Subjects and Resources and typically does not support making on-the-fly contextual decisions.
2. An RBAC-only approach can result in an extremely large number of roles in order to accomplish fine-grained authorization.
3. Resource Owners must know something about the roles and their intended purpose in order to accurately grant access to those roles.
4. In order to delegate rights against collections of resources, the resources must be organized into collections that facilitate delegation.
5. Given a very large number of roles and collections of resources, a correspondingly large number of delegations would have to be created and managed.

Benefits of ABAC

The benefits of ABAC are many and there are some requirements only achievable in an ABAC model. The most significant benefit is ABAC's user-intuitive nature which allows for an easy understanding of how a rule would grant access to a resource. RBAC seems foreign to many users and the levels of abstraction can be difficult for an IT team, especially during the early phase of its adoption. Other key benefits of ABAC include flexibility - almost anything can be represented as a rule-based query as long as the necessary data is available. A rule evaluated at runtime in a login session can make use of contextual information, even information passed in via SAML claims. In contrast, a stateless RBAC engine would not have access to this information when performing background pre-compilation of who has access to what.

ABAC Primary Benefits

1. ABAC makes it easy to specify access rules as simple queries.
2. ABAC rules can be extremely fine-grained and contextual.
3. ABAC rules can evaluate attributes of Subjects and Resources that are not inventoried by the authorization system.
4. ABAC rules need less maintenance and overhead because they don't require the creation or maintenance of the structure on which an RBAC model depends (e.g. roles and resource locations.)

Weaknesses of ABAC

One of the major challenges with ABAC is that the just-in-time evaluations of its rules result in a disconnect or lack of an auditable link between the rule-based policies and the resources (aka assets) that they protect. A simple example using a rule that grants access to an application based on attribute values and group membership highlights some of the weaknesses of ABAC. In this example, the rule allows all members of "Group C" where the user's Job Title in the HR system contains "sales" and their employee safety training status in the corporate training system is "Authorized" to access the application.

While this example illustrates the extensive flexibility of an ABAC-based approach in which rules can be created with almost any syntax desired and without all of the complex relationships that would be used in a similar RBAC assignment, the evaluation of the aforementioned rule, however, may not perform acceptably in real-time and may exceed a desirable time period. The challenge lies in how the application being protected is completely disconnected from the rule and from the conditions within the rule that specify group membership and attribute values. There is also a disconnect between the group used in the rule and the application to which it grants access—as the following shows.

Common audit questions arising from this scenario that would be extremely difficult to answer, if at all:

1. As an application owner, how would I see which users have access to my application and how they were granted this access? Do application owners need to be trained on how to read and interpret the rules as well as on how to research all the source information? How would the application owner determine who matches the rule, given that the source of the attribute values could come from another system or be contextual to a single login session? How would the application owner grant direct access to a user for the application in an

ABAC model?

2. As a delegated admin that can add or remove users from Group C, how would I know that adding users is granting them access to Application A? There is not a relationship between the group and the access it grants as there would be in an RBAC model.
3. As an auditor, how would I audit how access is granted to application and who is granting the access? Was it the person that added the user to Group C that granted the access to Application A or was it the training staff member that changed the user's status to "Authorized"? Who is accountable?
4. What security weight should be associated with the Job Title and training status fields on a user? From how many sources could this information be edited and by whom? Potentially a large number of people could be unintentionally performing "Entitlement Management" on a day to day basis.
5. In this scenario, how does an auditor monitor the transitions when users are granted and revoked access to sensitive applications dynamically? No log would be track persons who had access on a Monday versus a Friday because access would only be evaluated at runtime.

ABAC Primary Weaknesses

1. ABAC makes it extremely difficult, if not impossible, to determine permissions available to a particular user. Potentially, an extremely large number of rules might need to be executed, and in exactly the same order in which the system applies them, to successfully determine access. As a result, it could be impossible to determine risk exposure for any given employee position.
2. ABAC can lead to a "Rule Explosion" (somewhat in the same way as RBAC can create a "Role Explosion) as a system with N number of attributes would have 2^N possible rule combinations.
3. ABAC systems (which don't pre-calculate the net result of access rights) can be slow to answer authorization queries as well as limited in the sophistication of the rules resulting from ABAC's difficultly in accessing data from multiple source systems as needed and in an allotted time period.

Hybrid RBAC and ABAC Approaches

RBAC and ABAC are clearly two ways of approaching authorization problems and while both have overlapping qualities, each one individually cannot cover all the necessary aspects of access control.

For optimal, dynamic support of an IT organization's needs, systems supporting both RBAC and ABAC, such as EmpowerID, are necessary. The debate over which is the better access control system is not-applicable to *EmpowerID*'s Identity Management system because *EmpowerID* incorporates both role-based and attribute-based controls.

Types of RBAC/ABAC Hybrid Models and their strengths and weaknesses:

Attribute Centric

In an Attribute-Centric hybrid model, Roles are treated as just another attribute and permission sets are assigned to attribute-based policies. In contrast with conventional RBAC, a role isn't a collection of permissions but rather another attribute named "role." The main drawback to the Attribute-Centric model is the loss of RBAC's administrative ease when determining risk exposure for any employee position due to the lack of the relationship between the role and the access it grants when treated as another attribute.

Role Centric

In a Role-Centric hybrid model, permission sets are assigned to Roles and attribute-based policies are added to constrain RBAC. Constraint rules incorporating attributes can only reduce the permissions available users and not expand those permissions. Additionally, some of ABAC's flexible qualities are lost due to the fact that permission sets are still constrained by Role. The system, however, retains RBAC's qualities for determining the maximum set of permissions that are user-attainable.

Dynamic Roles

In a Dynamic-Roles model, permission sets are assigned to Roles just as they are in the Role Centric hybrid model. The Roles themselves however are assigned to users dynamically by attribute-based policies. This model supports the use of complex attribute-based permissions assignments, which automatically assign users to roles.

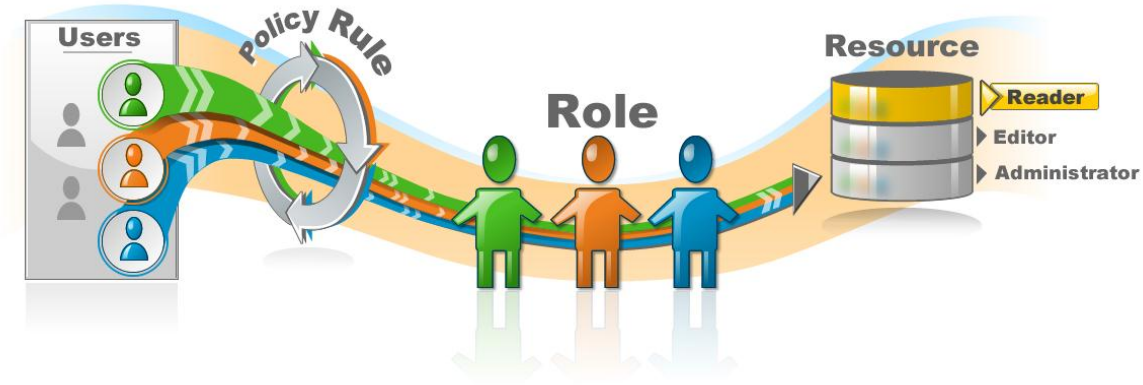

Figure 4. Dynamic roles where attribute-based policies automate role assignment

EmpowerID Combined RBAC/ABAC Model

EmpowerID optimizes RBAC-based authorization by leveraging the “Role-Centric” and “Dynamic Roles” approaches. EmpowerID is Role-Centric in its use of RBAC objects like roles for the bulk of its permissions assignments. Access granted to users based on their roles are still constrained and pared down at runtime in workflow processes by leveraging flexible rules that evaluate contextual information. This mix of RBAC and ABAC allows a structured approach to control who may do what with the additional ability to leverage adaptive or contextual security principles.

EmpowerID makes heavy use of Dynamic Roles to automate getting users into and out of roles based on information changes in key authoritative enterprise systems. Users can be automatically assigned to roles by falling into query-based collections known in EmpowerID as SetGroups. SetGroups are LDAP or code-based query collections of users and resources. SetGroups can leverage information from any reachable enterprise system to create and maintain these dynamic collections. By using SetGroups to control role membership, IT workload is dramatically reduced giving the benefits of an ABAC query-only model while maintaining RBAC’s ability to audit and review how access is being granted. SetGroups themselves are also RBAC actors in the EmpowerID model and can be directly assigned permissions as a type of dynamic role. SetGroups also automate the resource side of RBAC by creating dynamic attribute-based collections of resources to receive delegations. These collections of resources can use any metadata available for the resource allowing collections of secured resources to be grouped by security tagging or other information.

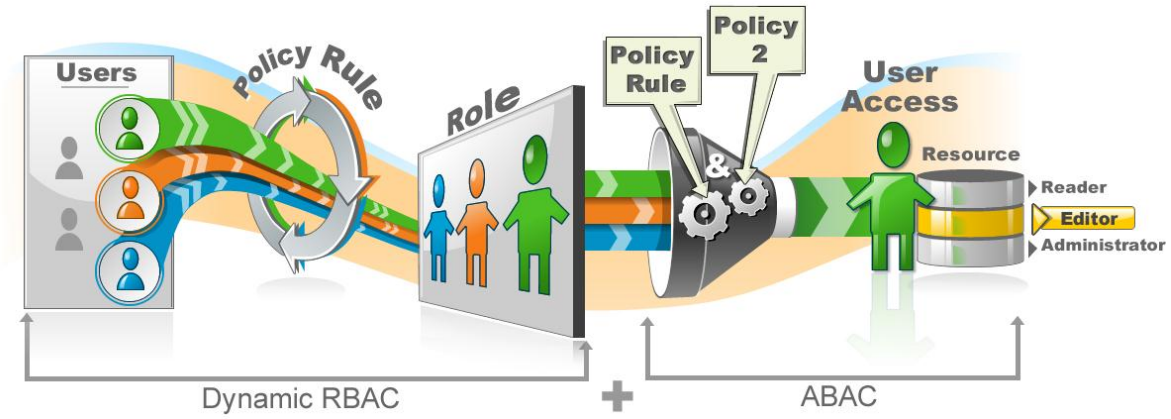

Figure 5. The EmpowerID hybrid RBAC and ABAC access control model

EmpowerID also supports true fine-grained ABAC in its SAML and WS-Trust Federation extensions. These are programmable extensions that can generate claims on the fly based on information from enterprise systems during the user login process for 3rd party applications that leverage claims for security.

An example of EmpowerID providing ABAC for 3rd party applications is its support for the role of Claims Provider for Microsoft SharePoint. SharePoint 2010 provides a pluggable model allowing authorization providers to be called during the user login process to add claims to a user's token. These EmpowerID claims are visible and can be used in the People Picker control in SharePoint to find and select people, groups, and claims when a site, list, or library owner assigns permissions. The claims for a user are evaluated at runtime during the login process allowing on the fly contextual authorization decisions to be made. These decisions can be as simple as leveraging a user's static assignments to roles and groups or something more complex where rule-based queries pull information from multiple enterprise systems to calculate access at that moment.

EmpowerID's combined hybrid model retains the advantages of RBAC while integrating the increased flexibility offered by ABAC approaches.

For more information on the *EmpowerID* authorization model visit:
http://wiki.thedotnetfactory.com/