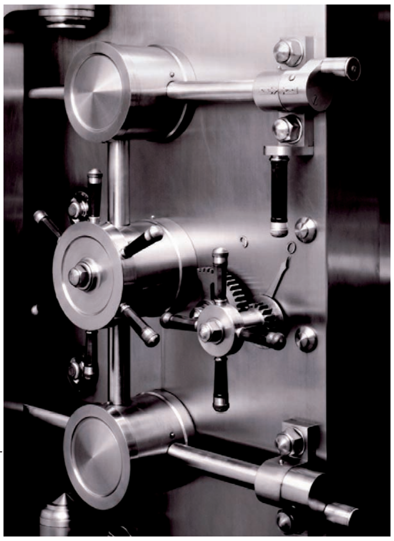

# ABAC and RBAC: Scalable, Flexible, and Auditable Access Management

**Ed Coyne,** *DRC*
**Timothy R. Weil,** *Coalfire*

As user populations of information systems have expanded, the challenge of controlling access to resources using security policies has grown. Researchers and system developers have simplified the administrative process by using groups of users who have the same authorizations. User groups were the precursor to *role-based access control.* RBAC groups permissions into roles and requires all access to occur through the RBAC system. Groups of permissions can then be readily provided to users in the simple operation of assigning roles. An enterprise's roles must be engineered to support security and business rules.

Over time, enterprises recognized a need for going beyond RBAC's groups of users and permissions. They needed to include attributes, such as time of day and user location, for distributed, dynamically changing systems. During this period, *attribute-based access control* was identified as a replacement for or adjunct to RBAC. ABAC uses labeled objects and user attributes instead of permissions to provide access control in a flexible manner. It was argued that ABAC could provide the flexibility needed in access control and that, if desired, RBAC could coexist with ABAC simply by considering a role as another attribute. Because ABAC doesn't use roles with permissions, it also avoids the need to engineer those roles and permissions. RBAC researchers have come up with several schemes for providing this attribute component—using constrained roles, for example.

## Role- vs. Attribute-Based Access

A certain simplicity in the ABAC idea is appealing. If a user has attributes that are reflected in the objects they want to access, then access is granted. On the other hand, with RBAC, the permissions granted to a user through roles must be evaluated to determine if the desired access will be granted. That is, a user is preassigned a set of roles (and thus permissions) with RBAC, while ABAC permissions can be acquired dynamically by virtue of the user's attributes. RBAC permissions are defined as an operation on an object, so only defined combinations of operations and objects are allowed. To achieve this granularity of access in ABAC requires rule sets that apply when attributes are evaluated.

When ABAC and RBAC are discussed together, the reasoning often goes like this:

- RBAC has been widely adopted and provides administrative and security advantages.
- However, it's outdated, expensive to implement, and unable to accommodate real-time environmental states as access control parameters.
- ABAC is newer, simpler to implement, and accommodates real-time environmental states as access control parameters.
- RBAC and ABAC can both be used by viewing roles as user attributes.

These statements are true and point toward using ABAC with role names as attributes. However, if this approach is taken, the result can be chaos.

RBAC is role-centric and ABAC is attribute-centric. Once roles become attributes, the advantages of RBAC are lost. Role names are still associated with users, but the consideration that roles are collections of permissions is no longer the case.

### Role-Based Access Control

With RBAC, roles can be well understood by their names, and they determine the sets of permissions to be granted to users. In addition, it's easy to audit which users have access to a given permission and what permissions have been granted to a given user. A limited number of roles can represent many users or user types, and roles can be assigned to users by non-expert personnel.

However, roles must be engineered before RBAC can be used. Furthermore, RBAC must be constrained to handle dynamically changing attributes, such time of day and location. Core RBAC can't handle such attributes.

### Attribute-Based Access Control

With ABAC, there's no need to engineer roles as long as role names aren't used as attributes. Dynamically changing attributes, such as time of day and location, can be accommodated in access control decisions. However, a potentially large number of attributes must be understood and managed, and attributes must be selected by expert personnel. Furthermore, attributes have no meaning until they're associated with a user, object, or relation, and it's not practical to audit which users have access to a given permission and what permissions have been granted to a given user.

### Implications

The downside of RBAC entailing a substantial role engineering effort is balanced by ABAC entailing a substantial attribute engineering effort. Furthermore, the perceived inability of RBAC to incorporate environmental attributes isn't present if constrained RBAC is used. ABAC can't audit user access to certain permissions, so RBAC with attributes is preferable to ABAC with role names as attributes.

## A Judicious Combination

In an earlier article,[1] we defined attribute-centric and role-centric access control models. Attribute-centric access control is where attributes control what resources a user can access. A role name (not a role, since a role has permissions in addition to its name) can be included in the attribute-centric model as one of the attributes assigned to a user. Thus, attribute-centric access control doesn't encompass RBAC, because permissions aren't included in the model.

In the role-centric access control model, roles with permissions determine what resources a user can access and how. Attributes can be added to RBAC to provide the flexibility needed in access control. Because existing RBAC models, defined in ANSI INCITS 359-2012 *Information Technology—Role Based Access Control*,[2] include constraints, it's an obvious solution to include attributes in RBAC by considering attributes to be constraints on access control decisions. In fact, this inclusion of attributes in RBAC as constraints has been written into a new standard.[3]

So what difference does it make whether we use an attribute-centric model (ABAC) or a role-centric model (RBAC)? Both seem to include attributes and roles. However, ABAC can contain role names only, not roles with their permissions. Therefore, to simulate RBAC, ABAC must include rules controlling the modes of access to the protected objects. In RBAC, the permissions explicitly define the modes of access.

Also pointed out in our earlier article is the fact that RBAC permits simplified auditing of the resources available to a given user as well as the users who have access to a given resource.[1] Auditing is accomplished simply by reviewing the roles available to a user, then enumerating permissions within this set of roles. Since the roles and permissions have been defined statically, a full enumeration of user-permission associations is easy to accomplish very quickly.

To accomplish this in ABAC requires an exhaustive enumeration of the attributes of a user and the corresponding attributes of the available protected objects. The full set of access rules, which could number in thousands in some cases, must then be instantiated with user and object attribute values. Because attributes can change dynamically, determining a user's potential permission set will also require instantiating rules with all possible attribute values while a user is active.

For example, if a user is currently on project A but also sometimes works on projects B and C, rules must be instantiated and evaluated with each of these three values. If the user has another attribute with three possible values (1, 2, or 3), then rules must be instantiated with nine possible value combinations for these two attributes. We quickly reach a combinatorial explosion of possible rule instantiations to evaluate: with $k$ attributes of $v$ values each, we'll need a set of $v^k$ rule evaluations. We again point out that with role name as only an attribute in ABAC, the auditing advantage of RBAC isn't present.

Conceptually, ABAC and RBAC are similar. Figure 1 illustrates this similarity. It is the properties of each model that give them their nature and behavior.

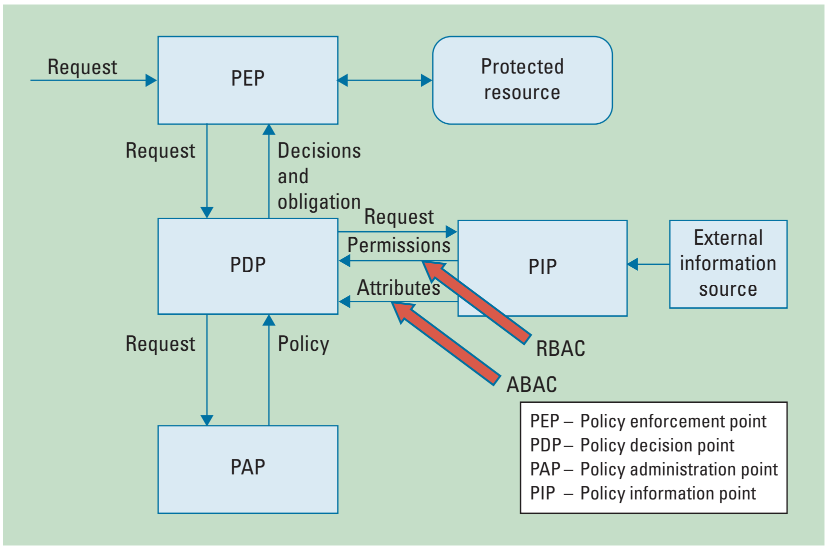
Request
PEP
Protected resource
Request
Decisions and obligation
PDP
Request
Permissions
Attributes
PIP
External information source
RBAC
ABAC
Request
Policy
PAP
PEP – Policy enforcement point
PDP – Policy decision point
PAP – Policy administration point
PIP – Policy information point

**Figure 1. Attribute-based and role-based access control. Permissions and attributes can take part in access decisions.**

As illustrated in Figure 1, combining ABAC and RBAC isn't an architectural challenge. Each model would have its own rule base in the policy information point (PIP). The policy decision point (PDP) would need the capability to evaluate these rules to produce an access decision.

Thus, it's possible to obtain the flexibility and advantages of ABAC while maintaining RBAC's advantages for analysis and risk control, if roles are used to define the maximum set of permissions that users can have. Clearly, the subject can't receive any permission not authorized for the active role or restricted by the attribute-based constraints. Permissions available to users in this approach therefore will be the intersection of *P* and *R*, where *P* is the set of permissions assigned to the subject's active roles and *R* is the set of permissions specified by the applicable ABAC rules. The user's role set therefore determines the maximum set of available permissions, supporting the principle of least privilege and allowing easy review of user permissions. Note that if *P* (the RBAC permission set) is all permissions, the system is equivalent to a "conventional" ABAC approach, where permissions are determined solely by attributes.

ABAC and RBAC, although similar, have particular advantages and disadvantages. When combined judiciously, the combination can provide access control that's scalable, flexible, auditable, and understandable. Significantly, current research in this topic includes the Role-Centric Attribute-Based Access Control (RABAC) work by Jin Xin and his colleagues,[4] which has realized one of the first reference models combining both roles and attributes in a reliable manner that preserves the best features of both access control methods.

Commercial implementations are also developing that use both role-centric and dynamic role capabilities combined with the features of ABAC's fine-grained authorization,[5] demonstrating that the approach defined by ANSI/INCITS 494-2012 is practical, and can combine the best features of RBAC and ABAC for the enterprise.

## References

1. D.R. Kuhn, E.J. Coyne, and T.R. Weil, "Adding Attributes to Role Based Access Control," *Computer*, vol. 43, no. 6, 2010; http://csrc.nist.gov/groups/SNS/rbac/documents/kuhn-coyne-weil-10.pdf.
2. *ANSI INCITS 359-2012 Information Technology—Role Based Access Control*, InterNational Committee for Information Technology Standards (INCITS), May 2012; www.techstreet.com/products/1837530.
3. *ANSI INCITS 494-2012 Information Technology—Role Based Access Control—Policy-Enhanced*, InterNational Committee for Information Technology Standards (INCITS), Aug. 2012; http://webstore.ansi.org/RecordDetail.aspx?sku=INCITS+494-2012.
4. J. Xin, R. Krishnan, and R. Sandhu, "A Role-Based Administration Model for Attributes," *Proc. 1st Int'l Workshop Secure and Resilient Architectures and Systems*, ACM, 2012, pp. 7–12.
5. "Best Practices in Enterprise Authorization: The RBAC/ABAC Hybrid Approach," white paper, EmpowerID, 2013; http://blog.empowerid.com/Portals/174819/docs/EmpowerID-WhitePaper-RBAC-ABAC-Hybrid-Model.pdf.

***Edward J. Coyne*** *is a senior security engineer in the High Performance Technologies Group at DRC. Contact him at ecoyne@drc.com.*

***Timothy R. Weil*** *is a senior security consultant at Coalfire. Contact him at tim.weil@coalfire.com.*