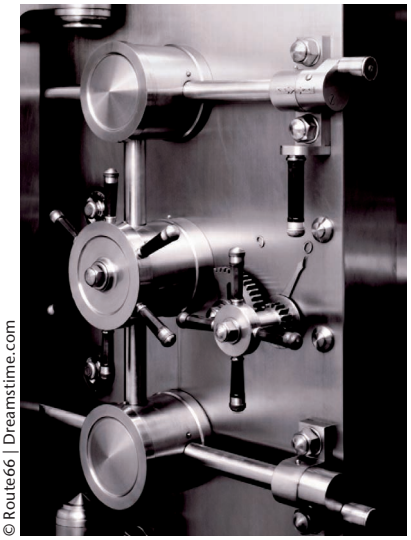

ABAC and RBAC:
Scalable, Flexible, and Auditable Access Management

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

 Published by the IEEE Computer Society