# Attribute-Based Access Control

W en.wikipedia.org/wiki/Attribute-Based_Access_Control

**Attribute-based access control** (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes etc.). This model supports Boolean logic, in which rules contain "IF, THEN" statements about who is making the request, the resource, and the action. For example: IF the requestor is a manager, THEN allow read/write access to sensitive data.[1]

Unlike Role-Based Access Control (RBAC), which employs pre-defined roles that carry a specific set of privileges associated with them and to which subjects are assigned, the key difference with ABAC is the concept of policies that express a complex Boolean rule set that can evaluate many different attributes.[2] Attribute values can be set-valued or atomic-valued. Set-valued attributes contain more than one atomic value. Examples are *role* and *project*. Atomic-valued attributes contain only one atomic value. Examples are clearance and sensitivity. Attributes can be compared to static values or to one another, thus enabling relation-based access control.

Although the concept itself existed for many years, ABAC is considered[3] "next generation" authorization model because it provides dynamic, context-aware and risk-intelligent access control to resources allowing access control policies that include specific attributes from many different information systems to be defined to resolve an authorization and achieve an efficient regulatory compliance, allowing enterprises flexibility in their implementations based on their existing infrastructures.

Attribute-Based Access Control is sometimes referred to as Policy Based Access Control (PBAC) or Claims Based Access Control (CBAC),[4] which is a Microsoft specific term. [5]

## Components

### Architecture

ABAC comes with a recommended architecture which is as follows:

1. The PEP or Policy Enforcement Point: it is responsible for protecting the apps & data you want to apply ABAC to. The PEP inspects the request and generates an authorization request from it which it sends to the PDP.
2. The PDP or Policy Decision Point is the brain of the architecture. This is the piece which evaluates incoming requests against policies it has been configured with. The PDP returns a Permit / Deny decision. The PDP may also use PIPs to retrieve missing metadata
3. The PIP or Policy Information Point bridges the PDP to external sources of attributes e.g. LDAP or databases.

### Attributes

Attributes can be about anything and anyone. They tend to fall into 4 different categories or functions (as in grammatical function)

1. Subject attributes: attributes that describe the user attempting the access e.g. age, clearance, department, role, job title...
2. Action attributes: attributes that describe the action being attempted e.g. read, delete, view, approve...
3. Resource (or object) attributes: attributes that describe the object being accessed e.g. the object type (medical record, bank account...), the department, the classification or sensitivity, the location...
4. Contextual (environment) attributes: attributes that deal with time, location or dynamic aspects of the

access control scenario[6]

### Policies

Policies are statements that bring together attributes to express what can happen and is not allowed. Policies in ABAC can be granting or denying policies. Policies can also be local or global and can be written in a way that they override other policies. Examples include:

1. A user can view a document if the document is in the same department as the user
2. A user can edit a document if they are the owner and if the document is in draft mode
3. Deny access before 9am

With ABAC you can have as many policies as you like that cater to many different scenarios and technologies. [7]

## Other models

Historically, access control models have included mandatory access control ( MAC), discretionary access control (DAC), and more recently role-based access control ( RBAC). These access control models are user-centric and do not take into account additional parameters such as resource information, relationship between the user (the requesting entity) and the resource, and dynamic information e.g. time of the day or user IP. ABAC tries to address this by defining access control based on attributes which describe the requesting entity (the user), the targeted object or resource, the desired action (view, edit, delete...), and environmental or contextual information. This is why access control is said to be attribute-based.

## Implementations

One standard that implements attribute- and policy-based access control is XACML, the eXtensible Access Control Markup Language. XACML defines an architecture, a policy language, and a request / response scheme. It does not handle attribute management (user attribute assignment, object attribute assignment, environment attribute assignment) which is left to traditional IAM tools, datatabases, and directories.

## Applications

### API & Micro Services Security

ABAC can be used to apply attribute-based, fine-grained authorization to the API methods or functions. For instance, a banking API may expose an approveTransaction(transId) method. ABAC can be used to secure the call. With ABAC, a policy author can write the following:

```
- Policy: managers can approve transactions up to their approval limit
- Attributes used: role, action ID, object type, amount, approval
limit.
```

The flow would be as follows:

1. The user, Alice, calls the API method approveTransaction(123)
2. The API receives the call and authenticates the user.
3. An interceptor in the API calls out to the authorization engine (typically called a Policy Decision Point or PDP) and asks: *Can Alice approve transaction 123?*
4. The PDP retrieves the ABAC policy and necessary attributes.
5. The PDP reaches a decision e.g. Permit or Deny and returns it to the API interceptor

6. If the decision is Permit, the underlying API business logic is called. Otherwise the API returns an error or access denied.

### Application Security

One of the key benefits to ABAC is that the authorization policies and attributes can be defined in a technology neutral way. This means policies defined for APIs or databases can be reused in the application space. Common applications that can benefit from ABAC are:

1. content management systems
2. ERPs
3. home-grown applications
4. web applications

The same process and flow as the one described in the API section applies here too.

### Database Security

Data-centric security for databases has long been specific to the database vendors: Oracle VPD, IBM FGAC, and Microsoft RLS are all means to achieve fine-grained ABAC-like security.

Using ABAC, it is possible to define data-centric policies that apply across multiple databases. This is called dynamic data masking.

An example would be:

```
- Policy: managers can view transactions in their region
- Reworked policy in a data-centric way: users with role == manager can do the
action == SELECT on table == TRANSACTIONS if user.region == transaction.region
```

### Big Data Security

Attribute Based Access Control can also be applied to Big Data systems like Hadoop. Policies similar to those used previously can be applied when retrieving data from data lakes.[8]

## References

1. ^ *"ABAC (Attribute Based Access Control), jerichosystems.com". Retrieved 2016-07-11.*
2. ^ *"SP 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations" (PDF). NIST. 2014. Retrieved 2015-12-08.*
3. ^ *"Attribute Based Access Control (ABAC), axiomatics.com". Retrieved 2016-07-05.*
4. ^ RBAC first – ABAC next, or what?, 2015, Horst Walther, *GenericIAM Blog*. Retrieved on 2016-08-30.
5. ^ Karp, Alan, Harry Haury, and Michael Davis. "From ABAC to ZBAC: the evolution of access control models." *International Conference on Information Warfare and Security*. Academic Conferences International Limited, 2010. Retrieved on 2016-08-30.
6. ^ http://stackoverflow.com/questions/36705901/alternatives-for-roles-claims-access-control-systems
7. ^ http://stackoverflow.com/questions/36705901/alternatives-for-roles-claims-access-control-systems
8. ^ http://www.prweb.com/releases/2016/10/prweb13771686.htm