Extensible Access Control Framework for Cloud based Applications

XACML v2.0 Attribute based Access Control (ABAC) Profile Version 1.0

19-02-2014
Version 1.0

Dr. Muhammad Awais Shibli
[Principal Investigator]

Dr. Arshad Ali
[Co-Principal Investigator]

National ICT R & D
[Funding Organization]
XACML v2.0 Attribute Based Access Control (ABAC) Profile Version 1.0

07 October, 2013

Editor:
Arjumand Fatima

Related Work:
This specification is related to
- eXtensible Access Control Markup Language (XACML) Version 2.0

Abstract:
This specification defines a profile for the use of XACML in expressing policies that use attributes based access control (ABAC) model.
## Extensible Access Control Framework for Cloud based Applications

KTH-SEECS Applied Information Security Lab, NUST-SEECS, H-12 Sector, Islamabad, Pakistan.  
Tel: 051-90852164, Fax: 051-8317363, Website: http://ais.seeecs.nust.edu.pk/project

### Contents

1. Introduction .................................................................................................................. 4
   - 1.1 Glossary .............................................................................................................. 4
   - 1.2 Related Terms .................................................................................................... 5
   - 1.3 Terminology ....................................................................................................... 5
   - 1.4 Normative References ....................................................................................... 5
   - 1.5 Non-Normative References ............................................................................. 5
   - 1.6 Background ....................................................................................................... 5
   - 1.7 Scope .................................................................................................................. 6
   - 1.8 Multiple Subjects ............................................................................................. 6
2. Example ....................................................................................................................... 6
   - 2.1 Example ABAC Policy ..................................................................................... 6
3. Assigning and Enabling Attributes ............................................................................. 8
4. Implementing the ABAC Model ............................................................................... 9
5. Profile ......................................................................................................................... 10
   - 5.1 Attributes ........................................................................................................ 10
   - 5.2 Attribute Assignment or Enablement .............................................................. 10
6. Conformance .............................................................................................................. 11
   - 6.1 As a policy processor ...................................................................................... 11
   - 6.2 As an XACML request generator ................................................................... 11
1. Introduction

{Non-normative}

This specification defines a profile for the use of the OASIS eXtensible Access Control Markup Language (XACML) [XACML] to meet the requirements for attribute based access control (ABAC) as specified in [NIST-ABAC]. Use of this profile requires no changes or extensions to standard XACML Version 2.0. This specification begins with a non-normative explanation of the building blocks from which the ABAC solution is constructed. A full example illustrates these building blocks. The specification then discusses how these building blocks may be used to implement the various elements of the ABAC model presented in [NIST]. Finally, the normative section of the specification describes compliant uses of the building blocks in implementing an ABAC solution. This specification assumes the reader is somewhat familiar with XACML.

1.1 Glossary

**ABAC**

“A logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes.” [NIST-ABAC]

**Attributes**

Predefined or pre-assigned characteristics which specify various aspects of the subject, resource, environment conditions, and/or requested actions. [NIST-ABAC]

**Subject**

An individual, process or device that represents an entity requesting to perform the operation upon the resource. [NIST-ABAC]

**Resource**

A logical object with an inherent value which requires to be protected by the unauthorized use by others. [NIST-ABAC]

**Access control**

A decision to allow or deny a subject access to a specific resource. [NIST-ABAC]

**Rule**

Specific allowable behavior based on the privileges of subjects and how resources are to be protected under which environment conditions. [NIST-ABAC]

**Action**

The execution of a function at the request of a subject upon a resource. [NIST-ABAC]

**Policy Decision Point (PDP)**

The system entity that renders an authorization decision based on the evaluation of one or more applicable policy(s). [XACML]

**Policy Enforcement Point (PEP)**

The system entity that makes decision request and enforces authorization decision in order to perform access control. [XACML]

**Policy Administration Point (PAP)**

The system entity that creates policies. [XACML]

**Policy Information Point (PIP)**

The system entity that provides attributes values for policy evaluation. [XACML]
1.2 Related Terms
Several closely related terms are used interchangeably in the Identity and Access Management domain. For ensuring clarity, this specification does not include certain of these terms. For instance, the term resource is used in place of object. Subject refers to requesters, initiators and users. Rule is used in place of permission, authorization, right, entitlement and privilege etc.

1.3 Terminology
The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

1.4 Normative References


1.5 Non-Normative References

1.6 Background
Attribute Based Access Control (ABAC) provides a highly flexible method for providing access based on the evaluation of attributes. ABAC is distinguishable from other logical access control models because it controls access to objects by evaluating rules against the attributes of the entities (subject and object) and the environment relevant to a request. ABAC systems are capable of enforcing both Discretionary Access Control (DAC) and Mandatory Access Control (MAC) models and can enable Risk-Adaptable Access Control (RadAC) solutions as well. An ABAC system can implement existing role-based access control policies and can support a migration from role-based to a more granular access control policy based on many different characteristics of the individual requester. ABAC offers the flexibility to apply access control policy without prior knowledge of the specific subject and for an unlimited number of subjects that might require access. One of the primary benefits of ABAC is its ability to accommodate external or new users without modifying the existing rules or object attributes as long as the subjects is assigned the attributes necessary for the access control decision.
Traditionally, logical access control models such as identity based access control (IBAC) or role based access control (RBAC) provided access of a resource to a locally identified subject or to locally defined roles that a subject has. Subject qualifiers such as identity and roles are often insufficient in the expression of real-world access control needs. In some cases, roles may not be sufficient for an access control decision and additional attributes may be required to be used in combination with the roles to render a decision. For example, the role “doctor” is not enough to decide whether he or she could perform a heart surgery. To perform a heart surgery a doctor must belong to cardiology department and have at least 5 years of consultation experience. These additional conditions can be easily catered by ABAC in terms of attributes whereas to model the same requirements in previous access control models added complexity and management problems at large scale. Additionally, these models usually offer static access control mechanisms with pre-defined or pre-determined privileges and do not easily support dynamic, multi-factor or multi-organizational authorization decisions. ABAC avoids the need for explicit authorizations to be directly assigned to individual subjects prior to a request to perform an operation on the object. A key feature of ABAC is the ability of the object owner to protect and share the object without any prior knowledge of individual subjects.

1.7 Scope

The policies specified in this profile do not answer the question “What set of attributes does a subject or a resource or an environment X have?” That question must be handled by an Attribute Assignment Authority, and not directly by an XACML PDP. Such an entity may make use of XACML policies, but will need additional information for attribute assignment.

The policies specified in this profile assume all the attributes for a given subject or resources have already been enabled at the time an authorization decision is requested. They do not deal with an environment in which attributes must be assigned or enabled dynamically based on the resource or actions a subject is attempting to perform.

1.8 Multiple Subjects

Attribute based access control may require policies to be applied on the action of more than one subject. Therefore, this profile supports the use of multiple subjects using XACML “subject-category”.

2. Example

This section presents a comprehensive example of the XACML policies associated with attribute based access control.

Consider the example of an educational institution which has students, faculty members and administrative staff in each department. Students can be enrolled in Undergraduate, Postgraduate or Doctoral programs. Faculty members can further have different designations such as Head of Department (HoD), Professor, Assistant Professor, Lecturer and Instructor etc. Administrative staff may have different organizational hierarchy. Access to any institutional resource is controlled based on the user’s role, designation and various other attributes. Sample usage of Attribute Based Access Control (ABAC) model in this scenario is given below.

2.1 Example ABAC Policy

The following plain-language rule is to be enforced:
A person can use the colored printer manufactured by HP for printing official documents during the working hours (i.e. 9.00am to 5.00pm) provided that he or she is the designated HoD from the department of Computing.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">printer</AttributeValue>
          <ResourceAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="false"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">color</AttributeValue>
          <ResourceAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:resource:type"
          DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="false"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">HP</AttributeValue>
          <ResourceAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:resource:manufacturer"
          DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="false"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="PrintRule" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">faculty</AttributeValue>
            <SubjectAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:subject:role"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
          </SubjectMatch>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">computing</AttributeValue>
            <SubjectAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:subject:department"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
          </SubjectMatch>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">HoD</AttributeValue>
            <SubjectAttributeDesignator AttributedId="urn:oasis:names:tc:xacml:1.0:subject:designation"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
    </Target>
  </Rule>
</Policy>
```
3. Assigning and Enabling Attributes

{Non-normative}

A subject may be assigned a set of subject attributes upon employment in an organization. Similarly, a resource may be assigned resource attributes upon its creation. Resource attributes may be embedded within the resources or stored and managed separately. Attributes and their values may then be modified throughout the lifecycle of subjects, resources and the attributes associated with them. As new subjects arrive in an organization, old subjects leave or characteristics of subjects change, the subject attributes must be updated. This needs to be done without
modifying the existing relationship between each and every subject and resource. Resource owners or administrators should be able to apply access control policies without prior knowledge of the specific subjects. To cater all these requirements, an implementation of ABAC model relies upon the prior assignment of subject attributes to subjects and resource attributes to resources.

The assignment and enablement of various subject, resource, action and environment attributes is outside the scope of the XACML PDP and are not discussed in detail in this profile. To perform this function, separate Attribute Enablement Authorities are required which can be implementation dependent. It is assumed that the attributes present in the XACML Request Context are already assigned to the given target element at the time the access decision is requested.

An Attribute Enablement Authority may deal with establishing and defining attribute values; constraining them and publishing schemas for them. In addition, it may offer methods for provisioning attributes. In some cases, the attributes may come from a digital repository containing all the attributes as well as the information regarding their association with a subject, resource, action or environment etc. This repository may respond to the attribute requests made by the PIP. Attribute Enablement Authorities MAY use an XACML Attribute Assignment <Policy> or <PolicySet> to determine whether an attribute can be assigned to a specific subject or resource. However, this is outside the scope of this profile.

4. Implementing the ABAC Model
{Non-normative}
The following sections describe how to use XACML policies to implement various components of the ABAC model as described in [NIST-ABAC].

ABAC model as defined in [NIST-ABAC] includes the following five basic data elements:

1. Users
2. Objects
3. Operations
4. Environment Conditions
5. Privileges

Users are implemented using XACML <Subjects> element. A Subject is often referred to as the user or requester. For the purposes of this document, it is assumed that the subject and user are synonymous, and the term subject is used throughout. Although a subject is used to represent both human and non-human entities which represent humans, for the sake of simplicity, the example in this document referred to subject as a human person.

Objects are implemented using XACML <Resources> element. A resource or object can be anything upon which an operation may be performed by a subject. Throughout this document, the term resource is used to represent the objects.

Operations are implemented using the XACML <Actions> element. An operation is the execution of a function at the request of a subject upon an object. Throughout this document, the term action is used to represent an operation.

Environment Conditions are implemented using the XACML <Environments> element.

Privileges are implemented using the XACML <Policy> or <PolicySet>. Policy, rules, and relationships govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to
be protected under which environment conditions. Throughout this document, the term policy is used to convey these rules and relationships.

The main functional points as defined in [NIST-ABAC] that provide services such as the retrieval and management of the policy, along with some logical components for handling the context or workflow of policy and attribute retrieval and assessment are as follows.

1. Policy Enforcement Point (PEP)
2. Policy Decision Point (PDP)
3. Policy Information Point (PIP)
4. Policy Administration Point (PAP)
5. Context Handler

Policy Decision Point (PDP) makes the access decisions by evaluating the applicable policies.

Policy Enforcement Point (PEP) enforces the policy decision for authorization in response to a request from a subject wanting to access a protected resource.

Policy Information Point (PIP) provides information needed by the PDP to make authorization decision.

Policy Administration Point (PAP) creates, thoroughly tests and manages the access control policies.

Context Handler converts authorization decisions in the canonical form (e.g., XACML) to the native response format and vice versa. It also manages the order in which policy and attributes retrieval and assertion is performed.

5. Profile

5.1 Attributes

Attributes SHALL be expressed using one or more XACML Attributes. Each application domain using this profile for attribute based access control SHALL define or agree upon one or more AttributeId values to be used for attributes. Each such AttributeId value SHALL be associated with a set of permitted values and their DataTypes. Each permitted value for such an AttributeId SHALL have well-defined semantics for the use of the corresponding value in policies.

5.2 Attribute Assignment or Enablement

An Attribute Enablement Authority, responsible for assigning attributes to subjects and resources and or enabling subject-resource relationships, MAY use an XACML Attribute Assignment <Policy> or <PolicySet> to determine which users or objects are allowed to have which attributes and under which conditions. There is no prescribed form for an Attribute Assignment <Policy> or <PolicySet>. It is RECOMMENDED that attributes in an Attribute Assignment <Policy> or <PolicySet> be expressed as Resource Attributes. It is RECOMMENDED that the action of assigning or enabling an attribute be expressed as an Action Attribute, where the AttributeId is action-id, the DataType is anyURI, and the <AttributeValue> is enableAttribute.
6. Conformance

An implementation may conform to this profile in one or more of the following ways.

6.1 As a policy processor

In order to conform to this particular specification as a policy processor, the implementation needs to make use of XACML policies.

6.2 As an XACML request generator

In order to conform to this particular specification as a request generator, the implementation needs to make use of XACML requests.