word account: 1340

**Investigation of The XACML Access Control Policy Language**

Jiashan Li

University of Calgary

**Introduction**

XACML (eXtensible Access Control Markup Language) is an approved OASISStandard access control language. XACML describes both an access control policy language and a request/response language implemented using XML. The policy language is used to express access control policies while the request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses).

**Design Rationale**

Most of the organizations are still using legacy system with in build authorization logics. Some times, one organization contains large number of information systems and applications that each system or application uses their own way of authorizing. Authorization has become more complex day by day. Because users within organization as well as outside the organization need access to shared data and collaborate efficiently. Therefore, it has been very hard to manage those legacy, custom authorization systems. So this is the reason why we have to use XACML language. XACML can provide a standardized, externalized and policy based approach for authorization. Also, it provides a fine-grained authorization with higher level of abstraction by means of policies and rules and supports dynamic evaluation of policies by using the Policy Information Point (PIP).

Here is the model of XACML. Subject (e.g., a human user or a program) wants to take some action on a particular resource. The subject submits its request to the entity protecting the resource (e.g., a file system). This entity is called a Policy Enforcement Point (PEP) which performs access control by making decision requests and enforcing authorization decisions. PEP forms a request to the Context handler in its native request format, optionally including attributes of the subject, resource, action, environment and other categories. Later, the context handler constructs a request using the XACML request language and sends it to the Policy Decision Point (PDP) which evaluates applicable policy and renders an authorization decision. The PDP obtains the appropriate policies (written in the XACML policy language) from the Policy Administration Point (PAP) and then it examines the request, retrieves policies that are applicable to this request, and determines whether or not access should be granted according to the XACML rules for evaluating policies. The answer (expressed in the XACML response language) is returned to the context handler, which can then allow or deny the access of the requester, by translating the response context to the native response format into the PEP. If access is permitted and obligations are fulfilled, then the PEP permits access to the resources, otherwise, it denies access.

**Major Features**

Compared with other access control policy language, XACML has some unique features. XACML is request centric. The policies in force are potentially applied to any information relating to that request. The fundamental issue is whether the access should be allowed. XACML allows other actions to be specified, but the focus if its design is about getting that yes or no answer.

Another unusual feature of XACML, which was discussed at length, is that all the information used by Policies is Attributes. Subjects, Resources, Actions and the Environment just have Attributes. There are no special Attributes that represent the "name" or "identity" of a Subject or a Resource. Attributes can be unique or not. In effect, XACML combines Identity and Capability models into one. This property of XACML is different from other Authorization systems, including SAML, which treats the Name of a Subject differently from other Attributes of that Subject.

The most useful feature of XACML is that Policies are dynamically bound to a Situation. Other access control systems, such as Permissions, ACLs, or ACIs statically associate the policy representation to the thing it applies to, usually a Resource, such as a file. Using XACML, the set of Attributes associated with the request determine what Policies apply. Two very similar requests might be controlled by completely different Policies.

**Access Control Paradigm and Use Case**

There is a use case which use the XACML model. A diamond company sells Diamond all other world. And it stores its sales figures in a database. There are many large number of employees in the Sale department across several regions and there are many large number of entries in the database as well. As this database contains highly sensitive data, the company wants to provide some access control on these data. It means that all employees are not able to access all data and only the authorized data must be accessed. Assume there are three employees in the sales department. The first one, Bob – Sale Person who is working on Asia region. He is assigned to SalePerson group. The second one, Alice – Region Sales Manager in Asia region. She is assigned to SaleManager group. The third one, Peter – Global Sales Manager. He is assigned to SaleManager role. And the method to use XACML access control model to limit the right of employees is divided into several steps.

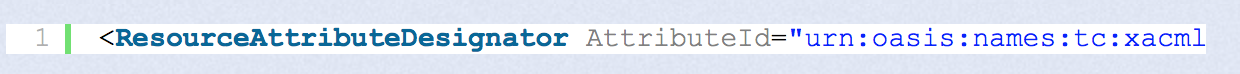
Firstly, Connect Identity Server in to the company user store. Then, add three policies in to PDP of Identity Sever based on XACML language. First, Access Policy – Sales data can be view by only the employees in Sale department. that is, employees who are assigned to SalePerson and SaleManager groups. Sales data can be only shared and deleted by Sale Managers. Second,Region Policy – Sales data related specific region must be read by who are in that region or who are in global sale managers. Third, Value Policy – If sales data related to more than $50 000 value, Sales person can not see the name of the client details. But Sale Managers can see.

Finally, download the company sample where all dependent libraries can be found inside lib directory in sample location. This sample contains a PEP data access module that modifies the data access requests. It is a simple module that extracts the Advice elements from XACML response and modifies the given requests. Once XACML response is received to the PEP data access module, it extracts input and output controllers from XACML Advice elements. Then PEP module can load the relevant request processor to modify the data access requests with input and output controllers. The PDP compares requests with policies and makes access control decision, then return back the result to let the user know whether they can get through.

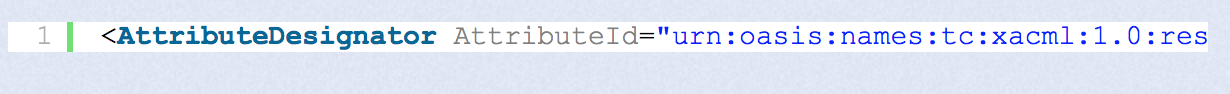
**Difference Between XACML 2.0 and XACML 3.0**

There is quantity of difference between XACML 3.0 and XACML 2.0, such as the introduce of Advice, the improvement in Target element, and there are more functions and algorithms in XACML. Here, we just take some of them to go through details.

Custom attribute categories can be defined with XACML 3.0. But in XACML 2.0, attributes have been organized into subject, resource, environment or action. According to the XACML 3.0 policy schema, category of XACML element is identified by a XML attribute called “Category”.  As an example, In XACML 2.0 Policy, you can define the attribute designator element as follows. But it must be a pre-defined category such as subject, resource, environment or action.



In XACML 3.0 Policy, you can define it as follows. Category can be any thing as it is defined as an attribute of AttributeDesignator element.



Another difference is improvements in obligation of XACML 3.0. For example, we want PEP to send an email to the user, In XACML 2.0, you need to define the obligation element with the user email statically. But user would not be same for each XACML request that is evaluated, Therefore, it is not possible to configure the email statically in the Obligation element. Obligation can only say PEP to “please send email to user”, however it doesn’t tell PEP who the user is and PEP has to figure out it by itself. This will cause a low efficiency. But in XACML 3.0, email of each user can be retrieved using PIP in a dynamic manner as we can define an expression element inside the ObligationExpression. Therefore, Obligation can say PEP to “please send email to user with a specific email address”.

**References**

1. “First Experiences Using XACML for Access Control in Distributed Systems.” Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, Sumit Shah.
2. eXtensible Access Control Markup Language (XACML) Version 3.0, January 22, 2013

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf>

1. stackoverflow

<http://stackoverflow.com/questions/7973277/how-does-xacml-3-0-differ-from-xacml-2-0>

1. Wikipedia

<https://en.wikipedia.org/wiki/XACML>

1. SOA Security

<http://soasecurity.org>

1. Unique Features of XACML

<http://xml.coverpages.org/XACML-PolicyModel.html>