XACML – Proposal for Policy Model Regarding Subject Semantics

October 16, 2001
V 0.2
Author: Michiharu Kudo

Status of this Document

This document is created to present to XACML policy model sub-committee a proposal of subject semantics. While the proposal focuses only subject semantics, the idea is easily applicable to object case as well.

1. Overview

This document proposes a definition of subject semantics in terms of XACML policy model. To illustrate the difference from subject semantics for XACML policy language, many examples are used in this proposal. In this proposal, policy model defines what a subject primitive means and what to specify as a subject. Policy language defines how to specify a subject. PDP specification defines how to retrieve subject information and all practical aspects. The proposal includes:

- In a policy model, subject semantics are defined as general as possible using logical expression based on attribute type-value pairs.
- In a policy language, subject semantics are represented in more application-specific ways as well as in original logical expression. It greatly facilitates readability of the access control rules.
- PDP specification presents how to retrieve subject related information into PDP and associate it to each language primitive.

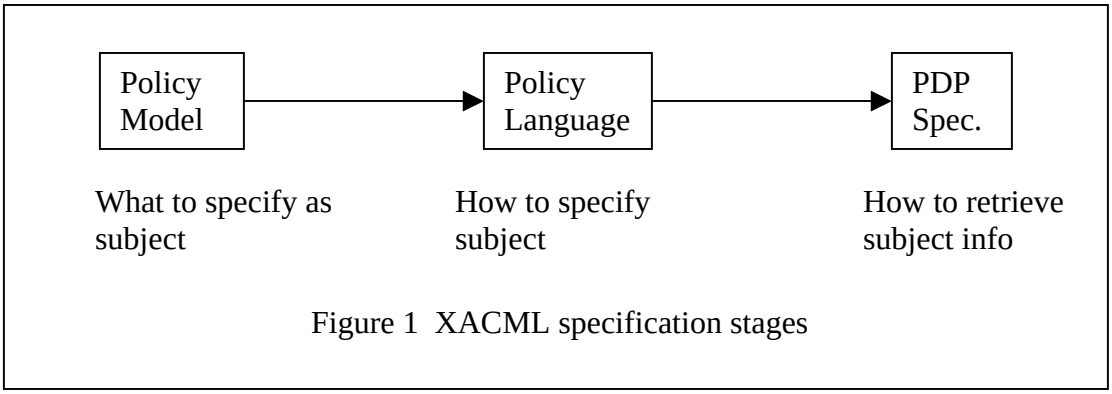

Figure 1 XACML specification stages

1.1 What Subject Should Mean

The “Subject” should mean only a subject who submitted an access request (Initiator in [ISO]). Other usage of the subject such as one to whom the authorization is granted should be distinguished from the usage as Initiator.

2. Subject Semantics in XACML Policy Model

Subject primitive in XACML policy model represents conditions on an Initiator (a requesting subject) for applying an access control rule. The semantics are represented as logical expression using Initiator’s attribute type-value pairs (conjunctive and disjunctive expression) since it is reasonable to represent conditions in general. In the policy model, we do not care about how to retrieve those information from outside but XACML PDP specification has to address such issue. For example, a subject who belongs to a marketing group, who is activating a manager role, and who holds a X.509 certificate issued from VeriSign is represented as Ex2-1:

Ex2-1: (group = ”marketing” AND role = ”manager” AND X509.Issuer = ”VeriSign”)
Ex2-2: ((group = ”development” AND role = ”manager”) OR group = ”marketing”)
Ex2-3: “*”

“*”, a wild card expression, implies every Initiator.

Above examples show subject conditions to be satisfied in applying a certain policy.

2.1 Underlying Policy Model

Underlying policy model for the above subject definition could be described as follows:

“In receiving a query, any access control rules which subject condition holds, object condition holds, action condition holds, and other conditions hold become candidate rules for making access decision”

“After every candidate rule is collected, PDP determines access decision (grant or deny, and additional conditions if necessary) according to the policy evaluation policy (it MAY include conflict resolution policy, default decision policy, decision propagation policy etc.)”

The above description shows that the logical expression for subject semantics makes sense since it is a generic condition specification method.

2.2 Subject Hierarchy With Wild Card Notation

In the policy model, it is desirable to deal with hierarchy regarding subject such as group hierarchy and role hierarchy. The proposal uses a text string representation that consists of hierarchy entities and a hierarchy separator just like directory path expression and Xpath expression. For example, the group

called LargeHospital under HealthcareIndustry department under Marketing organization can be represented as follows:

Ex2-4: (group = “/Marketing/HealthcareIndustry/LargeHospital”)

The above condition does not match to the LargetHospital group under Development organization. Since hierarchical semantics are encoded in the value of the group attribute, X.500 standard could be another possibility. Ex2-5 and 2-6 show wild card usages: every group under HealthcareIndustry department and every HealthcareIndustry department under some organization (e.g. development, service,) respectively.

Ex2-5: (group = “/Marketing/HealthcareIndustry/ *”)
Ex2-6: (group = “/ * /HealthcareIndustry”)

If a user belongs to “/Marketing/HealthcareIndustry/SmallHospital” group, then Ex2-5 matches. If a user belongs to “/Development/HealthcareIndustry” group, then Ex2-6 matches.

In handling above string representation, PDP will have to provide a group membership function that checks whether the group to which the subject belongs satisfies the hierarchical condition or not.

2.3 Inequality

Type-value pairs can be extended to support inequalities. For example, we may need to specify any user who does not belong to Administrator group. Ex2-7 shows an example:

Ex2-7: (group != “Administrator”)

We might need some discussion about whether we need a negation operator or not. The inequality covers a part of the negation semantics.

2.4 Other Entities Regarding Subject

The policy model may contain another subject information such as a person to whom the authorization is granted. If such information is used in the model, the policy model should provide a model primitive to represent such an entity apart from the said Subject. For example, Grantee primitive may be introduced. The semantics are represented in the same manner.

Ex2-8: (X509.Issuer = ”VeriSign” AND X509.dn = ”Alice”)

3. Subject Semantics in Policy Language

In general, we should at least provide a language primitive for logical expression that policy model defines. Examples are:

<formula>group=”marketing”</formula>

and

```
<formula type="equal"><type>group</type><value>marketing</value></formula>
```

3.1 Simpler Syntax

XACML should provide a simpler way for specifying subject attributes. In case of the group attribute for example, the following expressions are easier to read and shorter.

```
<group>marketing</group>
```

and

```
<subject group="marketing"/>
```

XACML should be as flexible as it allows each application to define local schema for the simple way of specifying subject (group element, role element, etc.) The above examples ignore specification details such as namespace prefix of XML Schema and standardized ways for specifying logical expression. Since it just shows a high level syntax of the language, XACML should consider policy language syntax more carefully.

3.2 Combination of Formula and Simpler Syntax

It would be more flexible to allow SSO (System Security Officer) to specify both the simpler syntax and the native formula expression at the same time. For example, Ex2-1 is specified differently as Ex3-1 shows. This facilitates both readability and flexibility.

Ex3-1:

```
<subject>
    <group>marketing</group>
    <role>manager</role>
    <formula type="equal"><type>X509.Issuer</type><value>VeriSign</value></formula>
</subject>
```

3.3 Conjunctive and Disjunctive Expression

It would be better to specify logical expression in more intuitive way. The following example represents Ex2-2 in different way assuming that each element within a subject element is conjunctively connected and subject elements in parallel are disjunctively connected.

Ex3-2:

```
<subject>
    <group>development</group>
    <role>manager</role>
</subject>
<subject>
    <group>marketing</group>
</subject>
```

The above expression is more intuitive than native formula expression below:

Ex3-3:

```
<subject>
  <formula type=”or”>
    <formula type=”and”>
      <formula type=”equal”><type>group</type><value>development</value></formula>
      <formula type=”equal”><type>role</type><value>manager</value></formula>
    </formula>
    <formula type=”equal”><type>group</type><value>marketing</value></formula>
  </formula>
</subject>
```

3.4 Wild Card Notation

“*” could be mapped to the following since “*” implies there is no conditions for subject. Note that this is different from the wild card usage in the subject path expression.

Ex3-4:

```
<subject>
  <all/>
</subject>
```

and

```
<subject/>
```

3.5 Inequality

When subject condition includes inequalities, policy language has to provide some ways to represent them. The following expression means that any user who belongs to the development group but not to the retired group.

Ex3-5: (group = ”development” AND group != ”retired”)

Ex3-5 can be represented using except element as Ex3-6 shows. Note that this example shows just a possible syntax.

Ex3-6:

```
<subject>
    <group>development</group>
    <except>
      <group>retired</group>
    </except>
</subject>
```

4. Subject Semantics in PDP Specification

In PDP specification, we should present how to retrieve subject related information (attributes, hierarchical relation, etc.) and associate it to each language primitive. Some of the subject information can be obtained from an access request. Other subject-related information may not be. Following is the possibility:

1. Access request contains subject-related information in SAML assertion
2. Access request contains subject-related information outside the SAML assertion
3. PDP asks PIP for subject-related information (attribute authority, LDAP, etc.)
4. PDP locally maintains subject-related information

Figure 2 illustrates relationship among authorization components. The first two items are contained in the access request in Figure 2. The third item is contained in the subject-related information from PIP. The last item is not drawn in the figure.

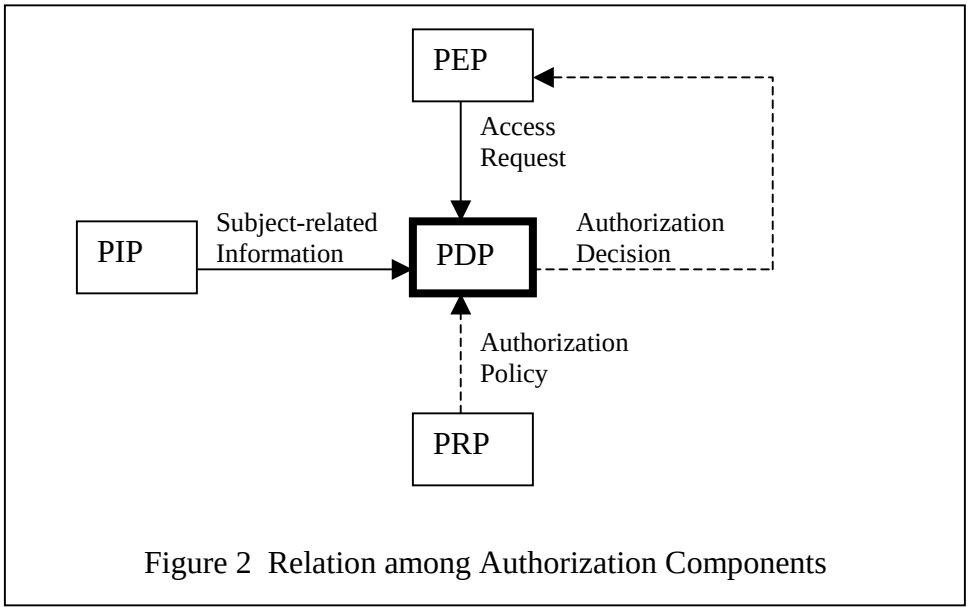

Figure 2 Relation among Authorization Components

Reference

[ISO] ISO/IEC 10181-3, Information technology – Open Systems Interconnection – Security frameworks for open systems: Access Control Framework , 1996.