XACML – Proposal for Policy Model Regarding Object Semantics

October 22, 2001
V 0.1
Author: Michihary Kudo

Status of this Document

This document is created to present to XACML policy model sub-committee a proposal of object semantics. This proposal is based on the semantics same as the subject case. The object can also be said as target and resource.

1. Overview

This document proposes a definition of object semantics in terms of XACML policy model. Since the base notion is the same as the proposal for the subject case, we removed the duplicated description.

1.1 What Object Can Mean

The "Object" can be used as a target resource of the access. According to the XACML charter description, the target of a policy (hereafter referred to as "target") can be any object that can be referenced using XML, that would be a directory, file, program, Web resource, servlet, EJB application, security label, category of the target, URI, XML document, XML fragment, DB table, view and record, etc.

2. Object Semantics in XACML Policy Model

Object primitive in XACML policy model represents conditions on an object for applying an access control rule. The semantics are represented as logical expression using object's attribute type-value pairs (conjunctive and disjunctive expression) since it is reasonable way 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, an object refered by URL like "http://www.xacml.org/marketing" is represented as Ex2-1. An object which is labeled as "confidential" security and "crypto" category or "privacy" security is represented as Ex2-2:

Ex2-1: (uri = "http://www.xcaml.org/xacml")
Ex2-2: ((sec_label = "confidential" AND category = "crypto") OR sec_label = "privacy")
Ex2-3: "*"

"*", a wild card expression, implies every Initiator.

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

2.1 Underlying Policy Model

Underlying policy model for the above object 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 object semantics makes sense since it is a generic condition specification method.

2.2 Object Hierarchy With Wild Card Notation

In the policy model, it is desirable to deal with hierarchy regarding object such as directory hierarchy. This 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 directory called system32 under winnt of c drive can be represented as follows:

Ex2-4: (path = "c:/winnt/system32")

Ex2-5 and 2-6 show wild card usages: every directories and files under winnt directory, and every directory that has system32 subdirectory as a child directory, respectively.

Ex2-5: (path = "c:/winnt/*")
Ex2-6: (path = "c:/ * /system32")

2.3 Inequality

Type-value pairs can be extended to support inequalities. For example, we may need to specify any files which does not have pdf extension. Ex2-7 shows an example:

Ex2-7: (file != "*.pdf")

3. Object Semantics in Policy Language

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

```
        <formula>uri="http://www.xcaml.org/xacml"</formula>
```
and
```
        <formula type="equal"><type>uri</type><value>http://www.xcaml.org/xacml
</value></formula>
```

3.1 Simpler Syntax

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

```
        <uri>http://www.xcaml.org/xacml</uri>
```
and
```
        <object uri="http://www.xcaml.org/xacml"/>
```

XACML should be as flexible as it allows each application to define local schema for the simple way of specifying object. 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:
```
  <object>
        <sec_label>not_restricted</sec_label>
        <category>standard</category>
        <formula>uri="http://www.xcaml.org/xacml"</formula>
  </object>
```

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 object element is conjunctively connected and object elements in parallel are disjunctively connected.
Ex3-2:
```
  <object>
        <sec_label>confidential</sec_label>
        <category>crypto</category>
  </object>
  <object>
        <sec_label>privacy</sec_label>
  </object>
```

The above expression is more intuitive than native formula expression below:
Ex3-3:
```
   <object>
      <formula type=”or”>
         <formula type=”and”>
            <formula
type=”equal”><type>sec_label</type><value>confidential</value></formula>
            <formula type=”equal”><type>category</type><value>crypto</value></formula>
         </formula>
         <formula type=”equal”><type>sec_label</type><value>privacy</value></formula>
      </formula>
   </object>
```

3.4 Wild Card Notation

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

Ex3-4:
```
   <object>
      <all/>
   </object>
```
and
```
   <object/>
```

3.5 Inequality

When object condition includes inequalities, policy language has to provide some ways to represent them. The following expression means that any files located under system32 which extension is not “dll”

```
         Ex3-5:   (directory = ”c:/winnt/system32” AND file != ”*.dll”)
```

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

Ex3-6:
```
   <object>
         <directory>c:/winnt/system32</group>
         <except>
            <file>*.dll</file>
         </except>
   </object>
```