XML Access Control Use Case of XACML

- Fine-grained XML Document Access Control -

July 25, 2002

Author: Michiharu Kudo

This is a proposal for how a fine-grained XML document access control could make use of XACML as its policy language. The fine-grained XML document access control means an element-wise (and an attribute-wise) access control in a XML document. Using fine-grained access decisions, an application can return only readable portions to the requesting subject. The policy specification is based on the latest XACML language proposal document [1].

1. Overview

Figure 1 shows a data-flow diagram for fine-grained XML access control.

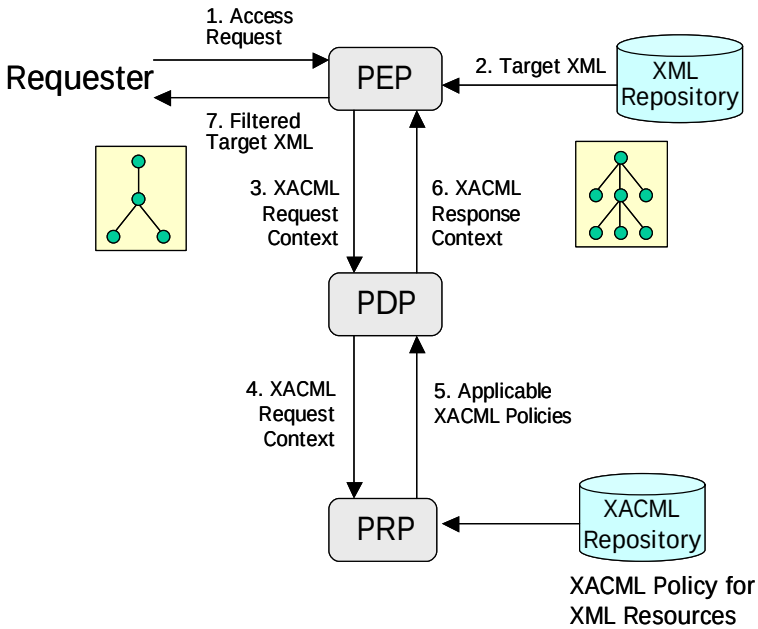

Figure 1 Data-flow diagram

Figure 1 shows an example data flow among XACML components (this diagram just indicates a logical relationship). XML documents are stored in XML repository. A requester submits an access request to PEP with a set of parameters including a target XML identifier (e.g. filename) and a

specific portion of the document (specific element, sub-tree, etc.) (step 1). That access request is converted to an XACML Request Context. The PEP also retrieves the target XML document from the repository and embed it in the XACML Request Context (step 2). We assume that this request context carries enough information to the PDP. After receiving the request context (step 3), PDP asks PRP to select a set of applicable XACML access control policies (step 4 and 5). Then the PDP makes decisions whether the requested portion of the document is readable or not. One or more access decisions (permit or deny on each element and attribute) called XACML Response Context are sent back to the PEP (step 6). Using those access decisions, the PEP filters no-readable elements (and attributes) out from the original document. The requester finally receives the XML document that contains only readable portions (step 7). Besides this scenario, it is easy to support another actions such as element-wise update and delete operations.

2. Example Document and Policy

2.1 Example Document

The following document is stored in XML repository as "A00.xml"

```
<a:employee xmlns:a="http://myNS">
  <a:name>Alice</a:name>
  <a:phone>111-1111</a:phone>
  <a:salary>10000</a:salary>
</a:employee>
```

2.2 Example Policy

Employees who belong to a "regular" employee group can read an employee element, a name element, and a phone element but cannot read a salary element. This access control policy is specified in XACML as follows:

```
<PolicyStatement PolicyId="PolicyForRegularEmployee" RuleCombiningAlgId="//.../PermitOverrides">
  <Description>Policy for regular employee group: employee, name, and phone elements are readable.</Description>
  <Target>
    <Subjects MatchId="function:string-equal" DataType="xs:boolean">
      <AttributeDesignator DataType="xs:string"
            Designator="/c:Request/c:Subject/c:SubjectAttribute/c:Attribute[@DataType='Group']/ text()"/>
      <Attribute DataType="xs:string">Regular</Attribute>
    </Subjects
    <Resources MatchId="function:string-match" DataType="xs:boolean">
      <AttributeDesignator DataType="xs:string"
            Designator="/c:Request/c:Resource/c:ResourceAttribute/c:Attribute[@DataType='XML']/ text()"/>
      <Attribute DataType="xs:string">A0[0-9]¥.xml</Attribute>
    </Resources>
    <Actions MatchId="function:string-equal" DataType="xs:boolean">
      <AttributeDesignator DataType="xs:string"
            Designator="/c:Request/c:Action[@Namespace=" urn:oasis:tc:XACML:action:XMLAC"]/text()"/>
      <Attribute DataType="xs:string">read</Attribute>
    </Actions>
  </Target>
  <RuleSet>
    <Rule RuleId="rule1" Effect="Permit">
      <Target>
        <Subjects MatchId="function:true"/>
```

```
            <Resources MatchId="function:node-equal" DataType="xs:string">
              <AttributeDesignator
                Designator="//c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute DataType="xs:string">//c:ResourceContent/a:employee</Attribute>
            </Resources>
            <Action MatchId="function:true"/>
          </Target>
        </Rule>
        <Rule RuleId="rule2" Effect="Permit">
          <Target>
            <Subjects MatchId="function:true"/>
            <Resources MatchId="function:node-equal" DataType="xs:string">
              <AttributeDesignator
                Designator="//c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute
                DataType="xs:string">//c:ResourceContent/a:employee/a:name</Attribute>
            </Resources>
            <Action MatchId="function:true"/>
          </Target>
        </Rule>
        <Rule RuleId="rule3" Effect="Permit">
          <Target>
            <Subjects MatchId="function:true"/>
            <Resources MatchId="function:node-equal" DataType="xs:string">
              <AttributeDesignator
              Designator="//c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute
                DataType="xs:string">//c:ResourceContent/a:employee/a:phone</Attribute>
            </Resources MatchId="function:true"/>
            <Action MatchId="function:true/>
          </Target>
        </Rule>
        <Rule RuleId="rule4" Effect="Deny">
          <Target>
            <Subjects MatchId="function:true"/>
            <Resources MatchId="function:node-match" DataType="xs:string">
              <AttributeDesignator
              Designator="//c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute
                DataType="xs:string">//c:ResourceContent/a:employee/a:salary</Attribute>
            </Resources>
            <Action MatchId="function:true"/>
          </Target>
        </Rule>
      </RuleSet>
    </PolicyStatement>
```

<PolicyStatement> contains a target specification and 4 rules. <Target> specifies applicability of this policy that the requester's group is regular, target XML resource matches "A0[0-9]¥.xml, and the action is read. The first rule compares "//c:ResourceContent/a:employee" element below the request context with the value of the node attribute specified in the request context (some node). If two elements are identical, then this rule returns "Permit". The second and third rules are interpreted similarly. The fourth rule compares "//c:ResourceContent/a:employee/a:salary" element with the value of the node attribute specified in the request context (some node). If two elements are identical, then this rule returns "Deny".

2.3 Example Request Context

We assume that the following request context is issued by the PEP.

```
<c:Request xmlns:c="urn:oasis:names:tc:xacml:context xmlns:a="http://myNS">
  <c:Subject SubjectCategory="urn:oasis:names:tc:xacml:identifiers:AccessSubject">
    <c:SubjectId Format="xs:string">Alice</c:SubjectId>
    <c:SubjectAttribute Name="Group" >
      <c:Attribute DataType="xs:string">Regular</c:Attribute>
    </c:SubjectAttribute>
  </c:Subject>
  <c:Resource>
    <c:ResourceSpecifier Format="XPointer" Scope="Immediate"
                         ResourceID="//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:name)"/>
    <c:ResourceAttribute Name="Location" >
      <c:Attribute DataType="xs:string">//org</c:Attribute>
    </c:ResourceAttribute>
    <c:ResourceAttribute Name="XML" >
      <c:Attribute DataType="xs:string">A00.xml</c:Attribute>
    </c:ResourceAttribute>
    <c:ResourceAttribute Name="Node" >
      <c:Attribute
          DataType="xs:string">//c:ResourceContent/a:employee/a:name</c:Attribute>
    </c:ResourceAttribute>
    <c:ResourceContent>
      <a:employee xmlns:a="http://myNS">
        <a:name>Alice</a:name>
        <a:phone>111-1111</a:phone>
        <a:salary>1000</a:salary>
      </a:employee>
    </c:ResourceContent>
  </c:Resource>
  <c:Action Namespace="urn:oasis:names:tc:XACML:action:XMLAC">read</c:Action>
</c:Request>
```

A subject element specifies a subject ID and a group attribute. A resource element specifies a target resource that is encoded using XPointer format. This data is given by the requesting application. In the resource ID, three types of data are encoded: an URI to the target XML document (//org), XML document file name (A00.xml), and the target portion of the document is "/a:employee/a:name", headed by "//c:ResourceContent" that navigates the correct location of the embedded XML document. Each parameter is extracted and three different resource attributes are generated. These parameters are referred from the policy using XPath. (See Section 4.2 if embedding the target XML in the context is not desirable.)

2.4 How to Generate Access Decision

Step 1: When PDP receives the above request context, it passes the context to PRP. The PRP tries to find a set of applicable policies that matches the request context. Since all target conditions specified in the policy statement (PolicyForRegularEmployee) are satisfied, the PRP returns that policy as an applicable policy.

Step 2: The PDP starts to evaluate the rule portion of the applicable policy. The first rule (rule1) compares two node sets: one that is obtained by applying an XPath expression obtained from the request context ("Node" resource attribute) on the target XML document (embedded in the context) and the other that is obtained by applying an XPath expression described in the rule1 (//c:ResourceContent/a:employee) on the embedded target XML document. If both node sets contain an identical node, then the target becomes true.

Step 3: If two node sets have no intersection, the node-equal function returns false. Then the target of the rule becomes false. In the case of the first rule (rule1), the target becomes false while the target of the second rule (rule2) becomes true. If the target becomes true, the rule combining algorithm (PermitOverrides) returns “permit” or “deny” depending on the value specified in the effect attribute. In the case of the rule2, it returns “permit”.

2.5 Example of Response Context

The following is an example of XACML Response Context saying that the requesting subject is allowed to read the name element.

```
<c:Response xmlns:c=”urn:oasis:names:tc:xacml:context”>
  <c:Result>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:name)</c:ResourceId>
  </c:Result>
</c:Response>
```

3. Resource Matching

From XML resource protection viewpoint, it is important to define which granularity of the XML resource can be protected using fine-grained access control policy. We consider two kinds of reference methods:

1. Resource matching using XPath expression
2. Resource matching using regular expression

If we choose the resource matching using XPath expression, we always need a requested target XML document. On the other hand, the resource matching using regular expression does not need the requested target XML document The difference between two methods is briefly described in Section 3.2.

3.1 Resource Matching using XPath Expression

A) Access to a specific node of a specific XML file

This context fragment shows a case that a requester needs to access a specific node of XML document.

```
<c:Resource>
  <c:ResourceSpecifier ResourceURI=”//org/A00.xml#xpointer(/employee/name)”/>
  <c:ResourceAttribute Name=”Location” >
    <c:Attribute DataType=”xs:string”>//org</c:Attribute>
  </c:ResourceAttribute>
  <c:ResourceAttribute Name=”XML” >
    <c:Attribute DataType=”xs:string”>A00.xml</c:Attribute>
  </c:ResourceAttribute>
  <c:ResourceAttribute Name=”Node” >
```

```
        <c:Attribute DataType="xs:string">/employee/name</c:Attribute>
      </c:ResourceAttribute>
    </c:Resource>
```

Target resource reference uses an XPointer syntax [2][1]. The string before "#" character indicates URI of the target XML resource and the string after "#xpointer" indicates XPath expression [3]. In this example, the URI is "//org/A00.xml" and the XPath is "/employee/name". In this case, this XPath refers to a name element below the employee element. When a namespace identifier is used, the XPointer expression becomes:

```
    <c:Resource>
      <c:ResourceSpecifier ResourceURI="//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:name)"/>
      <c:ResourceAttribute Name="Location" >
        <c:Attribute DataType="xs:string">//org</c:Attribute>
      </c:ResourceAttribute>
      <c:ResourceAttribute Name="XML" >
        <c:Attribute DataType="xs:string">A00.xml</c:Attribute>
      </c:ResourceAttribute>
      <c:ResourceAttribute Name="Node" >
        <c:Attribute DataType="xs:string">/a:employee/a:name</c:Attribute>
      </c:ResourceAttribute>
    </c:Resource>
```

The above specification means that each element in the A00.xml is accompanied with the namespace URI "http://myNS" which namespace prefix is "a". Note that in the example in Section 2, we added a prefix "/c:ResourceContext" before "a:employee/a:name" to make sure that the target path refers to the correct location of the request context.

The resource URI attribute can be an arbitrary URI syntax such as "file://c:/winnt/report.xml". In this case, the PDP and the PEP must have the identical meaning about the location of the target resource.

```
    <c:Resource>
      <c:ResourceSpecifier ResourceURI="file://c:/winnt/report.xml#xpointer(/overview)"/>
      <c:ResourceAttribute Name="Location" >
        <c:Attribute DataType="xs:string">file://c:/winnt</c:Attribute>
      </c:ResourceAttribute>
      <c:ResourceAttribute Name="XML" >
        <c:Attribute DataType="xs:string">report.xml</c:Attribute>
      </c:ResourceAttribute>
      <c:ResourceAttribute Name="Node" >
        <c:Attribute DataType="xs:string">/overview</c:Attribute>
      </c:ResourceAttribute>
    </c:Resource>
```

B) Access to a sub-tree in an XML document

There are some cases where the requester needs to access a specific sub-tree below a certain node not just a specific element. The PEP can tell that notion to the PDP by specifying "descendant" at the scope attribute in the resource specifier element. Then PDP returns multiple access decisions in

[1] XACML uses a subset of XPointer specification that allows only XPath expression in "#xpointer() function. The status of the latest XPointer specification is a candidate recommendation.

response to one access request. Note that the PDP must know the hierarchical structure of the target XML document.

```
<c:ResourceSpecifier Format="XPointer" Scope="descendant"
                     ResourceURI="//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee)"/>
```

The above context fragment is asking access decisions for every node below the employee element. The resultant response context includes four decisions as follows:

```
<c:Response xmlns:c=" urn:oasis:names:tc:xacml:context">
  <c:Result>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee)</c:ResourceId>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:name)</c:ResourceId>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:phone)</c:ResourceId>
    <c:Decision>deny</c:Decision>
    <c:ResourceId>//org/A00.xml#xmlns(a=http://myNS)xpointer(/a:employee/a:salary)</c:ResourceId>
  </c:Result>
</c:Response>
```

Since the salary element is not readable by the regular employee group, "deny" is returned for the salary node. By using this decisions, the PEP can create a requester's view that does not include a salary element:

```
<a:employee xmlns:a="http://myNS">
  <a:name>Alice</a:name>
  <a:phone>111-1111</a:phone>
</a:employee>
```

C) Access to a collection of XML document

There are cases where the target XML resource is not one XML document but a collection of the documents.

```
<c:Request xmlns:c=" urn:oasis:names:tc:xacml:context">
  <c:Subject SubjectCategory="urn:oasis:names:tc:xacml:identifiers:AccessSubject">
    <c:SubjectId Format="xs:string">Alice</c:SubjectId>
    <c:SubjectAttribute Name="Group" >
      <c:Attribute DataType="xs:string">Regular</c:Attribute>
    </c:SubjectAttribute>
  </c:Subject>
  <c:Resource>
    <c:ResourceSpecifier Format="XPointer" Scope="immediate"
                         ResourceURI="//org/records#xmlns(a=http://myNS)xpointer(/a:employee)"/>
    <c:ResourceAttribute Name="Location" >
      <c:Attribute DataType="xs:string">//org</c:Attribute>
    </c:ResourceAttribute>
    <c:ResourceAttribute Name="XML" >
      <c:Attribute DataType="xs:string">A00.xml</c:Attribute>
    </c:ResourceAttribute>
    <c:ResourceAttribute Name="Node" >
      <c:Attribute DataType="xs:string">/a:employee</c:Attribute>
    </c:ResourceAttribute>
  </c:Resource>
  <c:Action Namespace="urn:oasis:names:tc:XACML:action:XMLAC">read</c:Action>
</c:Request>
```

Above query assumes that the resource URI “http://org/records” means a collection of XML documents. If the resource URI means a collection of XML document, the PDP and the PEP must share the notation for the unique identifier of the XML document. For example, we assume here that there are two XML documents are meant by that URI, which is identified by emp1 and emp2. A response context becomes:

```
<c:Response xmlns:c=” urn:oasis:names:tc:xacml:context”>
  <c:Result>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/emp1#xmlns(a=http://myNS)xpointer(/a:employee)</c:ResourceId>
    <c:Decision>permit</c:Decision>
    <c:ResourceId>//org/emp2#xmlns(a=http://myNS)xpointer(/a:employee)</c:ResourceId>
  </c:Result>
</c:Response>
```

3.2 Resource Matching using Regular Expression

Potential problem of using XPath expression in the fine-grained XML document control is that the XML document is needed to evaluate the XPath expression. In the case that the target XML is huge or the communication channel is narrow, it is not wise to retrieve a target XML document at the evaluation time. Another issue is that the PEP does not want to disclose the requested XML document to the PDP. A simple path expression is useful for such cases. The simple path expression uses regular expression to match a string representation of the requested path expression and a string representation of the path expression in the policy.

```
<PolicyStatement PolicyId=”PolicyForRegularEmployee” RuleCombiningAlgId=”//.../PermitOverrides”>
  <Description>Policy for regular employee group: employee, name, and phone elements are readable.</Description>
  <Target>
    <Subjects MatchId=”function:string-equal” DataType=”xs:boolean”>
      <AttributeDesignator DataType=”xs:string”
            Designator=”/c:Request/c:Subject/c:SubjectAttribute/c:Attribute[@DataType=’Group’]/ text()”/>
      <Attribute DataType=”xs:string”>Regular</Attribute>
    </Subjects
    <Resources MatchId=”function:string-match” DataType=”xs:boolean”>
      <AttributeDesignator DataType=”xs:string”
            Designator=”/c:Request/c:Resource/c:ResourceAttribute/c:Attribute[@DataType=’XML’]/ text()”/>
      <Attribute DataType=”xs:string”>A0[0-9]¥.xml</Attribute>
    </Resources>
    <Actions MatchId=”function:string-equal” DataType=”xs:boolean”>
      <AttributeDesignator DataType=”xs:string”
            Designator=”/c:Request/c:Action[@Namespace=” urn:oasis:tc:XACML:action:XMLAC“]/text()”/>
      <Attribute DataType=”xs:string”>read</Attribute>
    </Actions>
  </Target>
  <RuleSet>
    <Rule RuleId=”rule1” Effect=”Permit”>
      <Target>
        <Subjects MatchId=”function:true”/>
        <Resources MatchId=”function:string-match” DataType=”xs:string”>
          <AttributeDesignator
            Designator=”/c:Request/c:Resource/c:ResourceAttribute/c:Attribute[@DataType=’Node’]/text()”/>
          <Attribute DataType=”xs:string”>/a:employee</Attribute>
        </Resources>
        <Action MatchId=”function:true”/>
      </Target>
    </Rule>
    <Rule RuleId=”rule2” Effect=”Permit”>
      <Target>
        <Subjects matchId=”function:true”/>
```

```
            <Resources MatchId="function:string-match" DataType="xs:string">
              <AttributeDesignator
                 Designator="/c:Request/c:Resource/c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute DataType="xs:string">/a:employee/(a:name|a:phone)</Attribute>
            </Resources>
            <Action MatchId="function:true"/>
          </Target>
        </Rule>
        <Rule RuleId="rule3" Effect="Deny">
          <Target>
            <Subjects MatchId="function:true"/>
            <Resources MatchId="function:string-match" DataType="xs:string">
              <AttributeDesignator
              Designator="/c:Request/c:Resource/c:ResourceAttribute/c:Attribute[@DataType='Node']/text()"/>
              <Attribute DataType="xs:string">/a:employee/a:salary</Attribute>
            </Resources>
            <Action MatchId="function:true"/>
          </Target>
        </Rule>
      </RuleSet>
    </PolicyStatement>
```

We assume that the following request context is submitted from the PEP.

```
    <c:Request xmlns:c=" urn:oasis:names:tc:xacml:context">
      <c:Subject SubjectCategory="urn:oasis:names:tc:xacml:identifiers:AccessSubject">
        <c:SubjectId Format="xs:string">Alice</c:SubjectId>
        <c:SubjectAttribute Name="Group" >
          <c:Attribute DataType="xs:string">Regular</c:Attribute>
        </c:SubjectAttribute>
      </c:Subject>
      <c:Resource>
        <c:ResourceSpecifier Format="URI" Scope="Immediate"
                             ResourceURI="//org/A00.xml?/a:employee/a:name)"/>
        <c:ResourceAttribute Name="Location" >
          <c:Attribute DataType="xs:string">//org</c:Attribute>
        </c:ResourceAttribute>
        <c:ResourceAttribute Name="XML" >
          <c:Attribute DataType="xs:string">A00.xml</c:Attribute>
        </c:ResourceAttribute>
        <c:ResourceAttribute Name="Node" >
          <c:Attribute DataType="xs:string">/a:employee/a:name</c:Attribute>
        </c:ResourceAttribute>
      </c:Resource>
      <c:Action Namespace="urn:oasis:names:tc:XACML:action:XMLAC">read</c:Action>
    </c:Request>
```

Above query has a simple path expression "/a:employee/a:name". This path expression matches the path expression written in the second rule "/a:employee/(a:name|a:phone)".

4. Notes

4.1 Namespace

In this document, we used a namespace prefix in XPath expressions but it assumes the prefix is associated with appropriate namespace URI in the target XML document (XACML Request Context). More strict way of specifying namespaces is to use namespace-URI() function and local-name() function. It makes XPath expression very complicated. For example,

/a:employee/a:name becomes /*[namespace-URI() = “http://myNS” and local-name()=”employee”]/*[namespace-URI() = “http://myNS” and local-name()=”name”].

The reason we used the prefix-based syntax is the readability of the policy. If we use the prefix-based syntax, someone who manages the authorization system must be sure about the trust relationship between PEP and PDP so that they can share both the correct namespace URI and the corresponding prefix. Otherwise, the namespace prefix and URI meant by the policy writer may differ from the URI that is handed by the PEP.

4.2 Separation of Context from target XML document

In this document, a request context is used to embed the target XML document. One of the merits of this scheme is that the access control policy only has to worry about one input document (request context) but not many. Any portion of the target XML document can be referred only using XPath expression. A downside would be that the whole XML document must be copied into the request context structure and the namespace specification makes XPath expression complicated. Alternative is to add some annotation to the attribute designator indicating that the XPath is for the request context or one for the target XML document.

```
<Resources MatchId=”function:node-equal” DataType=”xs:string”>
  <AttributeDesignator Type=”reference” RefApply=”context” Apply=”document”
       Designator=”ResourceAttribute/ Attribute[@DataType=’Node’]/text()”/>
  <Attribute DataType=”xs:string” Type=”value” Apply=”document”>employee/phone</Attribute>
</Resources MatchId=”function:true”/>
```

A type attribute indicates that XPath expression is specified as a value or a reference. The reference means that the XPath should be first applied to “RefApply” attribute as a reference. An apply attribute indicates that on which resource should XPath expression be applied. “context” and “document” means the request context and the target XML document, respectively.

5. Reference

[1] XACML, The OASIS eXtensible Access Control Markup Language (XACML), Committee Draft Version 15 July, 2002, http://www.oasis-open.org/committees/xacml

[2] XPointer, XML Pointer Language (XPointer) Version 1.0, W3C Candidate Recommendation, 11 September 2001, http://www.w3.org/TR/2001/CR-xptr-20010911.

[3] XPath, XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath.