eXtensible Access Control Markup Language

Graham Hughes
Brief description

• Dedicated language for specifying access control rules in XML
• OASIS industry standard
• Several goals: universal, definitive, easy to change, language easy to extend, policies should be composable
Explanation of goals

• Unified: the language should be able to express everything relevant for access control

• Definitive: all the rules for access control should be in an XACML policy, not in the server code

• Easy to change: changes to access rules should be relatively easy to make
Explanation, cont.

- Extensible: the language should be easy to extend with new primitives, data types, as needed for specific applications
- Composable policies: should be able to make one policy out of several preexisting smaller ones
• Spec offers outrageously baroque dataflow model

• Core idea is a request is submitted, it is acted upon by a policy which results in a decision, which is returned to the requestor
Request, response structure

• XML documents with their own rules

• Request is in essence, a bag of attribute-typed value pairs

• Some secondary structure in Subject, Resource, Action

• Response is in essence one of four values: Permit, Deny, Not Applicable, Indeterminate
Example request

<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            DataType="urn:oasis:names:tc:xacml:1.0:data-type:x500name"
            Issuer="www.medico.com" IssueInstant="2001-12-17T09:30:47-05:00">
            <AttributeValue>CN=Julius Hibbert</AttributeValue>
        </Attribute>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:attribute:role"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            Issuer="www.medico.com" IssueInstant="2001-12-17T09:30:47-05:00">
            <AttributeValue>physician</AttributeValue>
        </Attribute>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:attribute:physician-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            Issuer="www.medico.com" IssueInstant="2001-12-17T09:30:47-05:00">
            <AttributeValue>jh1234</AttributeValue>
        </Attribute>
    </Subject>
    <Resource>
        ...
    </Resource>
</Request>
Example request, decoded

{  "Subject ID": "CN=Julius Hibbert",
   "Role": "physician",
   "Physician ID": "jh1234" }

Example response

<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:1.0:context"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
    http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
  <Result>
    <Decision>NotApplicable</Decision>
  </Result>
</Response>
Policy structure

- Policies take information from requests and give answers
- Also some “environmental” data external to requests
- Organized hierarchically into PolicySets, Policies and finally Rules, combined using combining algorithms
Writing a policy

- Every level of the tree can have boundary conditions, which are inherited
  - Simple queries on attributes, plus XPath
  - Rules may additionally have arbitrary Boolean logic
  - Rules have an associated Permit or Deny
Writing a policy, cont.

- Request is a bag of attribute-value pairs, so attribute value requests get bags of values back

- Some minimal string matching, regular expressions, XPath, arithmetic, date comparisons, “set” union/intersection/subsets, higher order functions
Writing a policy, cont.

Lisp:
(> (attribute "urn:example:age" :only-one t) 18)

XACML:
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
    <SubjectAttributeDesignator AttributeId="urn:example:age"
      DataType="http://www.w3.org/2001/XMLSchema#integer"/>
    </Apply>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
    18
  </AttributeValue>
</Condition>
Combining rules

• Policies take several rules and combine the results

• Anything out of its boundary conditions is implicitly Not Applicable

• Combining is done through several algorithms

• Policy combination is essentially the same
Combining rules

- Four combining algorithms in spec
  - Deny overrides (any deny wins)
  - Permit overrides (any permit wins)
  - First applicable (First definitive answer wins)
  - Only one applicable (Indeterminate if more than one rule gives an answer)
A partial example

```xml
<PolicySet PolicyCombiningAlgId="...:deny-overrides">
  <Target>
    ...
  </Target>
  <Resources>
    <Resource>
      <ResourceMatch MatchId="...:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
          http://www.medico.com/schemas/record.xsd
        </AttributeValue>
        <ResourceAttributeDesignator AttributeId="...:target-namespace" DataType="http://www.w3.org/2001/XMLSchema#string" />
      </ResourceMatch>
    </Resource>
  </Resources>
  ...
</PolicySet>
```
Other things

• Obligations: things that have to be done in case the rule fires
  • E.g. email

• Policies can refer to other policies by reference

• Environmental attributes can have data not in the original request
Extending XACML

• Several avenues for extension
  • Extension is done by defining new URIs and referring to them
  • Functions, data types, combining algorithms