Extending XACML for Open Web-based Scenarios

Claudio A. Ardagna[1] Sabrina De Capitani di Vimercati[1]
Stefano Paraboschi[2] Eros Pedrini[1]
Pierangela Samarati[1] Mario Verdicchio[2]

(1) Università degli Studi di Milano, (2) Università degli Studi di Bergamo

W3C Workshop on Access Control Application Scenarios
Luxemboug, 17-18 November 2009

Motivation

- Open Web service systems receive access requests from remote parties to access Web services

- These systems may not have prior knowledge of users (relationships with authentication may change)

  $\Longrightarrow$ Need for access control based on properties/certificates

  $\Longrightarrow$ Need for interactive access control systems

  $\Longrightarrow$ Need for an expressive and simple access control solution applicable in practice

Goal and Contributions

Extending XACML (the most significant proposal for access control over the Web) for supporting the new access control paradigm needed in open scenarios

- depart from traditional authenticate/authorize approach (credential-based authorizations)
- support of abstractions
- provide access control authorizations with reasoning capability (recursive reasoning)
- communication of protection requirements while protecting access policy and related information (dialog management and interactive access control)

With a limited impact on the original XACML specification

Credential-based Authorizations

- Allow reference to digital certificates
- Allow fine-grained reference to properties they certify and to conditions about them
  - Attributes represent the content of the credentials (e.g., last name)
  - Metadata represent properties on the credentials (e.g., type)
- What users can do then depend on assertions (attributes) they can prove presenting certificates
- Access control can respond with requirements that the requester must satisfy to get access

Credential-based Authorizations – XACML

Credentials/Metadata are represented as a new XML schema

- Root element `certifications` contains one or more elements `certification` (class of certificates)
- Element `certification` defines a condition on metadata and has an attribute *id*
- Each element `certification` contains one or more alternative `group` elements representing restrictions on metadata

Attributes are treated like any other property in XACML

- Each occurrence of a certified attribute is translated into a XACML element `SubjectAttributeDesignator`
  - Attribute *AttributeId* is the attribute name
  - Attribute *Issuer* points to a credential

Credential-based Authorizations – Example

```
<certifications>
  <certification id="IT_IC">
    <group>
      <type>
        identity_card
      </type>
      <issuer>
        IT_Gov
      </issuer>
      <method>
        X.509
      </method>
    </group>
    <group>
      <type>
        passport
      </type>
      <issuer>
        IT_Gov
      </issuer>
      <method>
        SAML
      </method>
    </group>
  </certification>
</certifications>
```

Metadata

```
<Rule RuleId="ExampleRule" Effect="Permit">
  <Target/>
  <Condition
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply
      FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
      <SubjectAttributeDesignator DataType="XMLSchema#string"
        Issuer="urn:ext:cred-reference:IT_IC"
        AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:city-birth" />
      <AttributeValue DataType="XMLSchema#string">
        Milan
      </AttributeValue>
    </Apply>
    <Apply
      FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-less-than">
      <SubjectAttributeDesignator DataType="XMLSchema#integer"
        Issuer="urn:ext:cred-reference:IT_IC"
        AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:year-birth" />
      <AttributeValue DataType="XMLSchema#integer">
        1981
      </AttributeValue>
    </Apply>
  </Condition>
</Rule>
```

XACML policy with conditions on certified attributes

Abstractions

- Allow for the derivation of new concepts from existing ones
- Represent a shorthand by which a single concept represents a more complex one

Example

id_document (abstraction head) defined as an abstraction of credentials: {identity_card, driver_license, passport} (abstraction tail)

A policy that requires an id_document is satisfied by providing any of the three credentials

Abstractions – XACML

To manage abstraction specifications XACML is integrated with XQuery

- Abstractions are represented as a new XML schema
  - Root element `abstractions` contains one or more elements `abstraction`
  - Each element `abstraction` has an attribute *id* (abstraction head) and a set of equivalences in element `is` (abstraction tail)
- Abstractions can be embedded in XACML conditions via an XQuery invocation
  - An XQuery function takes in input an abstraction head and returns an abstraction tail

Abstractions – Example

```
<abstractions>
    <abstraction id="id_document">
        <is>
            <item>identity_card</item>
            <item>driver_license</item>
            <item>passport</item>
        </is>
    </abstraction>
</abstractions>
```

Abstraction definition

```
<certifications>
    <certification id="IT_ABBR">
        <group>
            <type>
                local:expand('id_document')
            </type>
        </group>
    </certification>
</certifications>
```

Abstraction-based metadata condition

Recursive Conditions

- Recursion can be exploited to specify conditions on data with a recursive structure (e.g., delegation, supervisor)

- Recursive reasoning is needed, for example:
  - for expressing policies based on chain of credentials
  - for supporting delegation

Recursive Conditions – XACML

- Like for abstraction, recursion is supported by integrating XACML with an XQuery engine
  - Recursive conditions defined via recursive XQuery functions
  - Recursive functions embedded and referenced in the policies (no changes to the language) to define policy conditions based on recursive concepts
  - Recursive functions take in input the XACML context, and produce new information to be used in policy evaluation

Recursive Conditions – Example

```
<context>
   <doctor id="1">
      <first_name>George</first_name>
      <last_name>Williams</last_name>
      <specialized>Surgery</specialized>
      <sex>M</sex>
      <supervisor/>
   </doctor>
   <doctor id="2">
      <first_name>Charles</first_name>
      <last_name>White</last_name>
      <specialized>Pediatric Surgery</specialized>
      <sex>M</sex>
      <supervisor>
         <doctorid>1</doctorid>
      </supervisor>
   </doctor>
   <doctor id="3">
      <first_name>Mary</first_name>
      <last_name>Wilson</last_name>
      <specialized>Pediatric Allergy</specialized>
      <sex>F</sex>
      <supervisor>
         <doctorid>1</doctorid>
      </supervisor>
   </doctor>
</context>
```

XACML Context

```
<Condition
  FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
  <SubjectAttributeDesignator
    DataType="http://www.w3.org/2001/XMLSchema#string"
    AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:doctor-id" />
  <AttributeSelector RequestContextPath=
    "local:getSupervisor(//doctor[@id=
    //patient[@id=urn:oasis:names:tc:xacml:2.0:attribute:patient-id]
    /doctorid])/@id"
    DataType="http://www.w3.org/2001/XMLSchema#string" />
</Condition>
```

XACML Recursive Condition

Dialog – 1

- The server may not have all the information it needs to decide whether or not an access should be granted
- The requester may not know which certificates she needs to present to a server to get access

  $\Longrightarrow$ Dialog management supports a new way of enforcing the access control process
- The server can communicate which information is needed to evaluate a policy
- Allows the requester to hand over only the necessary credentials

Dialog – 2

Issue to be addressed: communication of access control restrictions to be satisfied

- Safeguard privacy of the involved parties
  - avoid unnecessary release of certificates and information
  - avoid leakage of access control policies and information

  $\Longrightarrow$ Disclosure policies

- We distinguish five different disclosure policies. Each one potentially used independently in any condition appearing in an expression

Dialog – 3

Example: identity_card.age $>$ 18

- Condition: the condition can be fully disclosed as it is
  E.g., identity_card.age $>$ 18
- Predicate: only the information that a property needs to be evaluated with respect to a predicate can be released
  E.g., identity_card.age $>$
- Property: only the information that a property needs to be evaluated can be released
  E.g., identity_card.age
- Credential: only the information that there is a condition about a credential can be released
  E.g., identity_card
- None: nothing can be disclosed about the condition

Dialog – XACML

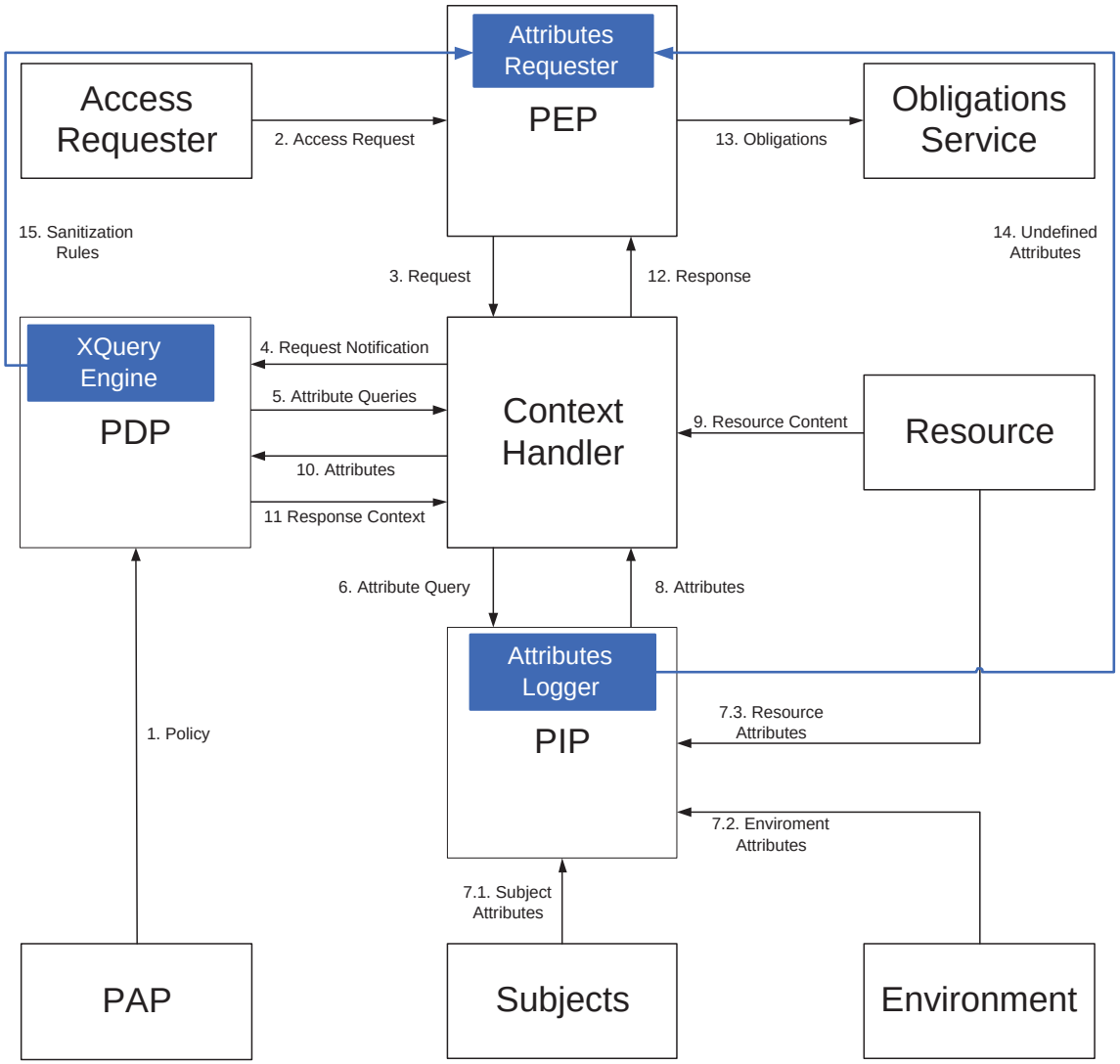
Dialog management requires a change in the XACML language

- Each condition in a XACML policy is associated with a disclosure policy
- A disclosure policy is represented with attribute *Disclosure* that is added to elements used for representing a condition
- Five possible values for attribute *Disclosure* corresponding to the five different disclosure policies:
  - condition
  - predicate
  - property
  - credential
  - none

Dialog – Example

```
<Rule RuleId="ExampleRule"  Effect="Permit">
  <Target/>
  <Condition
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply Disclosure="condition"
      FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
      <SubjectAttributeDesignator DataType="XMLSchema#string"
        Issuer="urn:ext:cred-reference:IT_IC"
        AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:city-birth" />
      <AttributeValue DataType="XMLSchema#string">
        Milan
      </AttributeValue>
    </Apply>
    <Apply Disclosure="predicate"
      FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-less-than">
      <SubjectAttributeDesignator DataType="XMLSchema#integer"
        Issuer="urn:ext:cred-reference:IT_IC"
        AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:year-birth" />
      <AttributeValue DataType="XMLSchema#integer">
        1981
      </AttributeValue>
    </Apply>
  </Condition>
</Rule>
```

Extended XACML Architecture



- PDP extended to support credential-based conditions, recursion, and abstraction (XQuery Engine)
- PIP enhanced to determine/store all the attributes that are not available at evaluation time and must be requested to the client (Attributes Logger)
- PEP extended with a communication channel to the PIP and PDP to retrieve the list of missing attributes, the conditions associated with them, and the disclosure policies (Attributes Requester)

Conclusions

- We presented possible extensions to the XACML language and architecture for fully supporting the requirements of an open Web-based scenario

- The extended XACML language and architecture support credential-based authorizations, abstractions, recursive conditions, and dialog management, with minimal impact on the standard