Toward Semantics-Aware Access Control

Ernesto Damiani
Sabrina De Capitani di Vimercati
DTI-University of Milan
damiani@dti.unimi.it
Outline

• Today:
  • XML-based languages
    • WS Security/Policy
    • XACML/SAML
    • Are they semantic-web ready?

• Tomorrow:
  • Semantics-aware access control
Web Security Issues

Related to Access Control
- How requestors are to be authenticated
- What qualifications a service consumer must prove to have in order to successfully access a Web Site/Service
- Which XML-based format/protocol is to be used for access-related information interchange

Related to Encryption
- Which XML elements are to be encrypted, using what algorithms and key sizes
- Which XML elements are to be integrity protected, using what mechanisms, with which algorithms and key sizes
Security Requirements

Authentication: WS Security, integration with Identity Management protocols (LDAP)

XML content-filtering, system authentication

Per-service & Per-user authorizations

Translation to legacy security mechanisms

Source: Vordel “Securing XML”
Policy language requirements

✓ A policy language must:
1. support access restrictions based on categorizations of users, purpose, types of operation, location, and data objects
2. be mobile (e.g., policies can travel together with data, or are sent on an alternate route)
3. express a variety of resources and actions on them
   ✓ Fine granularity on objects (e.g., conditions on individual interface items)
4. accommodate a variety of ways of identifying the owner of a privilege
   ✓ Selective release of subject information
Available XML-based AC languages

- **WS-Security/Policy** (Microsoft, IBM, SAP, BEA, Verisign)
- **XACML/SAML** (Oasis consortium with Sun, IBM, Entrust and others)
- Their XML schemata have been designed based on *models* describing the main entities (e.g., resource, subject) involved in AC
- **Non Semantic-Web aware**
  - Limited modelling of subjects
  - Limited modelling of resource types
Evaluating policies written in XML-based languages: a naive view

- `<requestHeader>`
  - `<credentials>`
    - `<credential value="foo" />`
  </credentials>
</requestHeader>

- `<context>`
  - `<communication protocol="HTTP" />`
</context>

- `<policy>`
  - `<resourceSelector>`
    - `<resource type="JPEG" />`
  </resourceSelector>
  - `<conditions>`
    - `<and>`
      - `<condition>RequestHeader/credentials/credential/@value = "foo"</condition>`
      - `<condition>Context/communication/@protocol = "HTTP"</condition>`
    </and>
  </conditions>
</policy>

Easy policy evaluation.. but sometimes one gets unexpected results

- **XML nodeset comparison is NOT string equality**
  - ✓ Type conversions may bring unexpected results, depending on source schemata

- **Another possibility: use policy-language specific elements-attributes for representing subject and object properties**
  - ✓ Need suitable mapping functions
WS Security – WS Policy suite

- Encompasses all aspects of WS security
- Version 2.0 available
- We will focus on WS-Security, WS-Policy and WS-Trust only
WS-Security

- XML-based **SIMPLE** mechanism to present credentials to Web services
  - Fully integrated with SOAP (credential are sent as headers)
  - Each SOAP request is authenticated either via username/password or via an X.509 certificate
  - Use of XML Signature and XML Encryption
  - Support for a wide variety of security token formats
    - Username (unsigned),
    - Kerberos, X.509 (signed)
  - Credentials may be encrypted while in transit
  - Selective association of encryption to services
    - AES 128 with token from X, 3DES with token from Y
Example

```xml
  <wsse:UsernameToken Id="MyID">
    <wsse:Username>Zoe</wsse:Username>
  </wsse:UsernameToken>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:SignatureMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
      <ds:Reference URI="#MsgBody">
        <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#MyID"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
</wsse:Security>
```
WS-Trust

✓ Main goals:
  ✓ Enable WS-based applications to exchange security tokens and manage trusted relationships
  ✓ Integrated with WS-Security
  ✓ Defines WSDL standard interfaces for:
    ✓ Security token creation, management and exchange
    ✓ Dissemination of credentials within different trust domains

• Main principles:
  ✓ Security token services form the basis of trust.
  ✓ Messages traveling on the Web MAY be required to prove a set of claims (e.g., name, key, permission, capability, etc.).
  ✓ Requestor MAY contact an appropriate authority (Security Token Service) which may require their own set of claims.
WS-Policy

- Flexible and extensible XML-based syntax for expressing fine-grained access policies to Web Services
- Single policy syntax for heterogeneous requirements (e.g. authentication, transport protocol selection, QoS, privacy)
- Relies on WS-Security to define subjects/objects
WS-Policy in a nutshell

- Conditions inside combinators (AND, OR, EXOR)
- Each condition may include a combinator or simple test on a value carried in the X.509 credential
- Express complex policies easily
- Fast evaluation
Policy Example

(1) <wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
(2)    <wsp:ExactlyOne>
(3)        <wsse:SecurityToken wsp:Usage="wsp:Required"
            wsp:Preference="100">
(4)            <wsse:TokenType> wsse:Kerberosv5TGT
                        </wsse:TokenType>
(5)        </wsse:SecurityToken>
(6)    <wsse:SecurityToken wsp:Usage="wsp:Required"
            wsp:Preference="1">
(7)        <wsse:TokenType> wsse:X509v3
                        </wsse:TokenType>
(8)    </wsse:SecurityToken>
(9)    </wsp:ExactlyOne>
(10) </wsp:Policy>
SOAP Request Example

(1) <S:Header>
(2) <wsse:Security ...>
(3)  <wsse:BinarySecurityToken
      ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary">
      MIIEZzCCA9CgAwIBAgIQEmtJZc0...
(4)  </wsse:BinarySecurityToken>
(5)  </wsse:BinarySecurityToken>
(6)  <wsse:BinarySecurityToken
      ValueType="wsse:Kerberosv5TGT"
      EncodingType="wsse:Base64Binary">
      JYTVjkvkjaOJK76i7tuaeHJ...
(7)  </wsse:BinarySecurityToken>
(8)  </wsse:BinarySecurityToken>
(9)  <wsse:UsernameToken >
(10)  <wsse:Username>edamiani</wsse:Username>
(11)  </wsse:UsernameToken>
(12)  </wsse:Security>
(13) </S:Header>
(14) <S:Body>
...
(15) </S:Body>
WS-Policy issues

✓ Not very human-readable
✓ Some inconsistencies:
  <SecurityToken TokenType="UsernameToken" Usage="Required">
    ...
  </SecurityToken>

  <SecurityToken Usage="Required">
    <TokenType>UsernameToken</TokenType>
    ...
  </SecurityToken>

✓ Unclear semantics
  <MessageAge Usage="Ignored" Age="3600" />
SAML rationale

- SAML (Security Assertion Markup Language): XML-based syntax for exchanging security data across heterogeneous, distributed systems boundaries
- Major design goal: Single Sign-On (SSO)
- SAML assertions are declarations of fact, according to some authority
- SAML assertions are compounds of one or more of three kinds of “statement” about a “subject” (human or program):
  - Authentication
  - Attribute
  - Authorization decision
- Assertions can be digitally signed
SAML Request information

- A SAML Request provides either a query or a request for a specific assertion.

- Parameters of a query could be:
  - Issuer ID and timestamp
  - Assertion ID
  - Subject
    - Name plus the security domain
    - Optional subject confirmation, e.g. public key
  - "Conditions" under which assertion is valid
    - SAML clients must reject assertions containing unsupported conditions
    - Special kind of condition: assertion validity period
XACML

✓ XACML (EXtensible Access Control Markup Language): XML-based syntax for expressing fine-grained access policies. Independent of SAML specification.
  ✓ General-purpose authorization policy model and XML-based specification language for heterogeneous, corporate-wide policies.
✓ Triple-based policy syntax: <Object, Subject, Action>
  ✓ Negative authorization supported
✓ Input/output to the XACML policy processor defined as XACML “context” data structure
✓ Input data referred by XACML-specific attribute designators as well as XPath expressions
✓ Extension points: function, identifier, data type, rule-combining algorithm, policy-combining algorithm, etc.
✓ A set of policies can be combined by a higher level policy (policySet statement)
XACML rules and policy

XACML defines three top-level policy elements:

- **Rule** element contains a boolean expression that can be evaluated in isolation
  - It is not intended to be accessed in isolation by a Policy Decision Point (*PDP*). It is not intended to form the basis of an *authorization decision* by itself.
  - It is intended to exist in isolation only within an XACML Policy Administration Point (*PAP*), where it may form the basic unit of management, and be re-used in multiple policies.

- **Policy** element contains a set of <Rule> elements and a specified procedure for combining the results of their evaluation.
  - It is the basic unit of policy used by the *PDP*, and so it is intended to form the basis of an authorization decision.

- **PolicySet** element contains a set of *Policy* or other *PolicySet* elements and a specified procedure for combining the results of their evaluation.
  - It is the standard means for combining separate *policies* into a single combined policy.
Representing knowledge about subjects and resources

- XACML provides:
  - facilities to support different characteristics of the subject:
    - identity
    - role [RBAC].
  - a standard way to reference the attributes defined in LDAP
    - This is intended to encourage implementers to use standard attribute identifiers for some common subject attributes.
Conditions based on resource content

• In many applications, it is required to base an authorization decision on data contained in the information resource to which access is requested.
  ✓ For instance, a common component of privacy policy is that a person should be allowed to read records for which he or she is the subject. The corresponding policy must contain a reference to the subject identified in the information resource itself.

• XACML provides facilities when the information resource can be represented as an XML document.
  ✓ The `<AttributeSelector>` element may contain a XPath expression over the request context to identify data in the information resource to be used in the policy evaluation.
Overview of XACML Core Schema

```xml
<Policy>
  <Target>
    <Resources>
      <Subjects>
        <Actions>
          <RuleSet ruleCombiningAlgId = "DenyOverrides">
            <Rule ruleId="R1">
              <Rule ruleId="R2">
                ...
                <Obligations>
                <RuleSet>
            </Rule>
          </RuleSet>
        </Actions>
      </Subjects>
      <Resources>
      <Target>
        <Resources>
        <Subjects>
        <Actions>
          <RuleSet ruleCombiningAlgId = "DenyOverrides">
            <Rule ruleId="R1" Effect="Permit">
              <Target>
              <Resources>
              <Subjects>
              <Actions>
                <Condition>
                </Rule>
              </Condition>
            </Rule>
          </RuleSet>
        </Actions>
      </Subjects>
      <Resources>
          <Rule ruleId="R2" Effect="Deny">
            <Target>
            <Resources>
            <Subjects>
            <Actions>
              <Condition>
              </Rule>
            </Condition>
          </Rule>
        </Actions>
      </Subjects>
      <Resources>
      </Target>
    </Policy>
```
Policy Example

```xml
<Policy PolicyId="identifier:example:SimplePolicy1">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="identifier:example:SimpleRule1" Effect="Permit">
    <Target>
      <Subjects><Subject>
        <SubjectMatch MatchId="function:rfc822name-equal">
          <SubjectAttributeDesignator AttributeId="identifier:subject:subject-id"
            DataType="identifier:datatype:rfc822name"/>
          <AttributeValue
            DataType="identifier:datatype:rfc822name">*@medico.com</AttributeValue>
        </SubjectMatch>
      </Subject>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
    </Target>
  </Rule>
</Policy>
```
XACML Request/Response
Context Example (I)

```xml
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subjectid"
      DataType="identifier:rfc822name">
      <AttributeValue>bs@simpsons.com</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="identifier:resource:resource-uri" DataType="xs:anyURI">
      <AttributeValue>http://medico.com/record/patient/BartSimpson</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="identifier:example:action" DataType="xs:string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
</Request>

<Response>
  <Result>
    <Decision>Deny</Decision>
  </Result>
</Response>
```
Three key aspects of knowledge representation in XML languages for AC

• *Resource representation.* Writing access control policies where resources to be protected are pointed at via identifiers and access conditions are evaluated against their attribute values is not sufficient anymore.

• *Subject identity.* Evaluating conditions on the subject requesting access to a resource means accessing personal information either presented by the requestor as a part of the authentication process or available elsewhere. A number of alternatives to strong identities are coming of age, all of them involving advanced metadata.

• *Context representation.* Distributed environments increase the amount of context information available at policy evaluation time (e.g., location-based info).
Yesterday’s metadata: PICS

- Platform for Internet Content Selection
- www.w3.org/PICS/
- Recommendation from 1996
- Web Site/pages rating
  - PICS labels within HTTP headers or META tags
    - from a document server or an independent label bureau
  - Electronically signed and associated to a digest
Today’s metadata: RDF/RDFS

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <rdf:description about="http://www.crema.unimi.it/Home/Damiani">
    <s:Creator>Ernesto Damiani</s:Creator>
  </rdf:description>
</rdf:RDF>
```

- **Resource**
- **namespace**
- **Domain vocabulary** = Defined using its own syntax, RDF Schema
- **property – value pair**
RDF Schemata = Ontologies

- List class and properties that can appear in RDF assertions about subjects and resources
- Define structured domain vocabularies
  - Specify relations between classes, including is-a, part-of
  - Constraints based on values
- Shared by actors in a given domain
Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:Class rdf:ID="Staff">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
</rdf:RDF>
```
Using ontology-based metadata in AC languages: three approaches

- **Specify access control requirements about resources in terms of metadata describing them.**
  - **Offline approach:** use metadata at policy writing time as a guideline. E.g. Available metadata about document ‘foo.dat’ say it is a business report -> suggest the policy allows the CEO to read it
  - **Online approach:** use metadata at policy evaluation time E.g.: The CEO can read all files whose (trusted) metadata say they are business reports
  - **Inverse approach:** use metadata as a guide to validate existing policies. Files “foo1.dat” and “foo2.dat” are both business reports, according to the associated metadata. Why are they treated differently by the policy? (C.Farkas et al., 2003)
Extending XACML to support RDF

**Three extension points:**

- Extend the XACML Context to include metadata associated with both subjects and resources.
- Extend the AttributeValue XACML element (used in XACML to qualify both subjects and objects) capability of specifying auxiliary namespaces. Auxiliary namespaces to be added are at least two:
  - the rdf: one, allowing for using RDF assertions as values for the XACML AttributeValue element
  - others (md: and ms:) enabling using properties and class names from a user ontology within those assertions.
- Extend the MatchID attribute by introducing a new function, called metadataQuery, expressing the processing needed for policy enforcement.
Sample RDF metadata associated with a SMIL presentation

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:md="http://ourdomain.it/MD/Schema/md-syntax#"
  xmlns:ms="http://ourdomain.it/MS/Schema/ms-syntax#">
  <rdf:Description
    rdf:about="http://ourdomain.it/MD/PresSMIL/presentation7318.smi">
    <rdf:type rdf:resource="http://ourdomain.it/MD/Schema/md-syntax#PresSMIL" />
    <md:title>Conference presentation 2004-02-13</md:title>
    <md:duration>7256992</md:duration>
    <md:format>application/smil</md:format>
    <md:contains>
      <rdf:Bag>
        <rdf:li rdf:resource="http://ourdomain.it/MD/Text/transcript7318.txt"/>
        <rdf:li rdf:resource="http://ourdomain.it/MD/Video/video010234.avi"/>
      </rdf:Bag>
    </md:contains>
  </rdf:Description>
</rdf:RDF>
```

**AC Policy:**
Trainers of the Teaching Quality Evaluation group are allowed to see SMIL presentations containing a video that shows trainers instructing trainees.
Comments

• The policy is composed of two assertions:
  • the type of user who can access the resource (Trainers of the Teaching Quality Evaluation group)
  • the kind of resources involved (SMIL presentations including a video that show trainers instructing trainees).

• Such assertions are used to select the target of the XACML rule based on metadata associated to resources.
Policy evaluation

When a policy involving metadata needs to be evaluated, the subject context already contains the RDF description of the requester. The policy evaluation engine works as follows:

- **First**, the semantic assertions about the requestor that are included in the subject field of our policy rules and the metadata about the requestor in the access request are compared to identify the policy rules that apply to the requestor.

- **Second**, the semantic assertions that are included in the resource context of applicable policy rules are used to query the descriptive metadata of the requested resource, checking whether the requested resource satisfies the rules selected in the previous step.
Policy evaluation (ctd.)

- Both these selection steps involve RDF queries, where the assertions in the policy rules are used to query metadata associated with the requester and the involved resource.

- Such querying can be tackled by means of two different techniques:
  - reasoning based on metadata
  - database-like querying.
Evaluating a request

- Consider now a request to see presentation 7318.smi submitted by a user who presents to our system some signed metadata stating that the requestor is Sam, an instructor trainer of the Teaching Quality Evaluation Department.
- Suppose also that according to the hierarchical organization of the concepts denoted by the domain ontologies, the ontology supports the specialization: "Instructor is a sub-class of Trainer".
- According to this specialization, the evaluation of the access request should return a permit decision because both Sam and the presentation involved in the request satisfy the subject and resource conditions specified in the rule.
Problems

- Although proposed extensions to XACML rely on standard RDF syntax, there are many ways to say the same thing in RDF.
- Some precautions must be taken to keep the computational complexity of enforcement under control.

✓ in our work, we prescribe that attribute values written in RDF use a RDF reification technique.
Example

Suppose a request comes in whose encapsulated metadata are:
(A, type, statement)
(A, subject, thisRequestUser)
(A, predicate, type)
(A, object, Trainer)

Then all XACML rules R whose subject metadata include (\(?\), subject, Trainer) (or its subtype (\(?\), subject, Instructor) ) will be selected.

Assume that the resource metadata mentioned in the context of the policy rule R is the following:
(\(?\), type, Statement)
(\(?\), subject, PresSMIL)
(\(?\), predicate, contains)
(\(?\), object, Video)

These metadata can now be used to build a query on the resource descriptors, to identify the objects to which the rule applies (e.g., the policy will apply to the SMIL presentation with the metadata shown before).

The reified statement contained in the policy is used to construct the query which is submitted to the set of resource descriptors.
References


P. Ceravolo, Managing Identities via Interactions between Ontologies. OTM Workshops 2003: 732-740

Other related papers:
Security Lab, Dipartimento di Tecnologie dell’Informazione (DTI), Milan: http://seclab.dti.unimi.it/

Upcoming forum:

2004 ACM Workshop on Secure Web Services (SWS)
October 29th, 2004
George W. Johnson Center at George Mason University, Fairfax, VA, USA
in conjunction with the Eleventh ACM Conference on Computer and Communications Security
Thank you for your attention!
An extended XACML policy

<?xml version="1.0" encoding="UTF-8"?>

<Rule
xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi= http://www.w3.org/2001/XMLSchema-instance
xmlns:ctx="urn:oasis:names:tc:xacml:1.0:context"
xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
xmlns:md="http://ourdomain.it/MD/Schema/md-syntax"
xmlns:ms="http://ourdomain.it/MS/Schema/ms-syntax"
RuleId="urn:oasis:names:tc:xacml:examples:ruleid:1"
Effect="Permit">
<Target>

-------- THE NEXT THREE SLIDES ---------------------

</Target>
</Rule>
<Subjects>

<Subject>

<SubjectMatch>
MatchId= "urn:ourdomain:function:metadataQuery">
<AttributeValue
DataType="http://">
<rdf:Statement rdf:about="thisRequestUser" >
<rdf:subject rdf:resource="http://ourdomain.it/MS/Schema/ms-syntax#Trainer" />
<rdf:predicate rdf:resource="http://ourdomain.it/MS/Schema/ms-syntax#belongs"/>
<rdf:object rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Teaching Quality Evaluation
</rdf:object>
</rdf:Statement>
</AttributeValue>
</SubjectMatch>
</Subject>
</Subjects>
<Resources>

<Resource>
  <ResourceMatch MatchId="urn:ourdomain:function:metadataQuery">
    <AttributeValue
      DataType="http://">
      <rdf:Statement rdf:about="thisRequestUrl">
        <rdf:subject rdf:resource="http://ourdomain.it/MS/Schema/md-syntax#PresSMIL"/>
          video"/>
        <rdf:object rdf:nodeID="content"/>
      </rdf:Statement>
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:ourdomain:attribute:metatag"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
</Resource>

<Resource>
  <ResourceMatch MatchId="urn:ourdomain:function:metadataQuery">
    <AttributeValue
      DataType="http://">
      <rdf:Statement rdf:about="thisRequestUrl">
        <rdf:subject rdf:resource="http://ourdomain.it/MS/Schema/ms-syntax#Trainer"/>
        <rdf:predicate rdf:resource="http://ourdomain.it/MS/Schema/ms-syntax#instructs"/>
        <rdf:object rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          Trainee
        </rdf:object>
      </rdf:Statement>
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:ourdomain:attribute:metatag"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
</Resource>

</Resources>