Toward Semantics-Aware Access Control

Ernesto Damiani
Marco Cremonini
DTI-University of Milan
[cremonini, damiani]@dti.unimi.it

Our Agenda

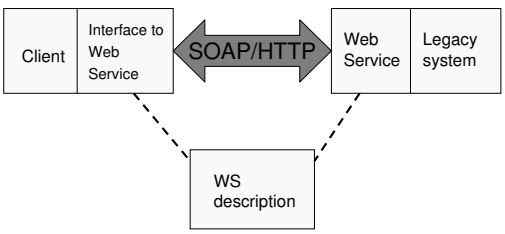
- ⌘ Web Services Reminder
- ⌘ Security issues
- ⌘ WS Security/Policy
- ⌘ XACML/SAML
- ⌘ Architectural patterns
- ⌘ Semantics-aware access control
- ⌘ XML Gateway

Web Services Reminder

What are Web Services?

- ✓ An XML-based technology for B2B integration and inter-company workflows
- ✓ Relies on the WWW proven infrastructure and protocols
- ✓ Integrate legacy applications
  - ✓ Low programming efforts/costs
  - ✓ Rapid development
- ✓ Main advantages: flexibility, lower costs, reduced time to market

WS Structure



WS 3-Step Process

1. Package software functionalities as a Web Service
   - ✓ Use a SOAP/XML wrapper
2. Publish service descriptions on the Net
3. Dynamically assemble new applications from services
   - ✓ Use a directory to locate them at run-time

WS Pillars

- ✓SOAP: Parameters encoding, services API (application programming interface)
- ✓WSDL: services descriptions
- ✓UDDI: directory of services

WS Pillars: SOAP

- ✓Simple Object Access Protocol
  - ✓Messaging protocol for remote procedure call
  - ✓Alternative to RPC, RMI, Corba, DCOM, etc.
- ✓Technology:
  - ✓XML messages
  - ✓Transport via HTTP, SMTP or other protocol

WS Pillars: WSDL

- ✓Web Services Description Language
- ✓Describes:
  - ✓Service Interface
  - ✓Transport Protocol
  - ✓Payload Messages
- ✓Used by development tools and runtime environment to publish service interfaces

WS Pillars: UDDI

- ✓Universal Discovery Description and Integration registry
  - ✓Open or protected searchable directory
  - ✓Describes business objects and their services
- ✓White and Yellow Pages :
  - ✓Publication and discovery of services
  - ✓Identification, authentication, authorization of client applications

UDDI Directory Contents

- ✓Business Information
  - ✓Name, Description
- ✓Business Service
  - ✓Category
- ✓Service Specifications
  - ✓Technology Model (tModel) identifiers
  - ✓WSDL interface
- ✓Access info: protocol, address, description

WS Platforms (I)

- ✓Application Servers:
  - ✓Microsoft .NET
    - ✓IIS
  - ✓Java 2 Enterprise Edition (J2EE)
    - ✓IBM WebSphere
    - ✓BEA Web-Logic
    - ✓Sun SunONE
    - ✓Oracle 9i Application Server
    - ✓JBoss (open source), etc.

WS Platforms (II)

	.NET	J2EE
Vendor	Microsoft	BEA, HP, IBM, Sun, Oracle
Operating System	Windows	Unix/Linux/Win
Languages	C#	Java
Database	SQL Server	Oracle, DB2, etc.

WS Open Issues

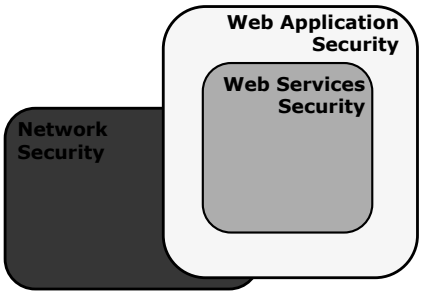
- ✓ Lack of design skills and implementation experience
- ✓ Performance
- ✓ Reliability, QoS
- ✓ Transaction integrity support
- ✓ Orchestration/Workflow management support
- ✓ **Security**

Security Concerns

⌘"45,5% of IT managers considered security their biggest Web Services concerns" – Business Week, 2002

⌘"By 2005, XML based communications using SOAP will have reopened 70% of the attack paths against Internet-connected systems, which were closed by network firewalls in the 1990s" – Gartner Group, October 2002

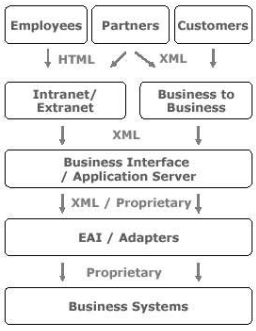
The Big Picture



Security Questions

- ✓ Related to Access Control
  - ✓ How requestors are to be authenticated
  - ✓ What qualifications the service consumer must demonstrate in order to successfully access a Web Service
- ✓ 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: SSL/WS Security, Integration with Identity Management (LDAP)

XML content-filtering, System authentication

Per-service & Per-user authorization

Translation to legacy security mechanisms (RACF)

Source: "Securing XML", Vordel

WS Access Control

- ✓Wanted:
  - ✓An AC model and a policy language suitable for Web Services
  - ✓An enforcement architecture, examining incoming SOAP requests and evaluating policies to grant or refuse access to the WS
  - ✓An efficient reference implementation

Requirements (I)

- ✓ Policy language must support access restrictions based on categorizations of users, purposes, types of operation, and data objects
- ✓ Policy could be mobile (e.g., together with data, or sent on an alternate route)

Requirements (II)

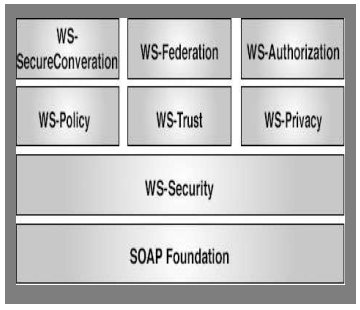
- ✓ Policy language must express a variety of resources and actions on them
- ✓ Policy language must accommodate a variety of ways of identifying the owner of a privilege

Language/Model

- ✓Requirements
  - ✓Fine granularity on objects (e.g., conditions on individual interface items)
  - ✓Selective release of subject information
- ✓Competitors:
  - ✓WS-Security/Policy (Microsoft, IBM, SAP, BEA, Verisign)
  - ✓XACML/SAML (Oasis consortium with Sun, IBM, Entrust and others)

WS Security/Policy

WS Security – WS Policy standard suite



- ✓ Encompasses all aspects of WS security
- ✓ Still preliminary
- ✓ 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

Goals

Goals:

- ⌘ Multiple security tokens for authentication or authorization
- ⌘ Multiple trust domains
- ⌘ Multiple encryption technologies
- ⌘ End-to-end message-level security and not just transport-level security

Non-Goals:

- ⌘ Establishing a security context or authentication mechanisms that require multiple exchanges.
- ⌘ Key exchange and derived keys
- ⌘ How trust is established or determined.

Main Features

- ✓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 I

```
1 <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
2     <wsse:UsernameToken Id="MyID">
3         <wsse:Username>Zoe</wsse:Username>
4     </wsse:UsernameToken>
5     <ds:Signature>
6         <ds:SignedInfo>
7                 <ds:SignatureMethod
                          Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
8                 <ds:Reference URI="#MsgBody">
9                         <ds:DigestMethod
                                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
10                        <ds:DigestValue>LyLsF0Pi4wPU...</ds:DigestValue>
11                </ds:Reference>
12        </ds:SignedInfo>
13        <ds:SignatureValue>DJbchm5gK...</ds:SignatureValue>
14        <ds:KeyInfo>
15                <wsse:SecurityTokenReference>
16                        <wsse:Reference URI="#MyID"/>
17                </wsse:SecurityTokenReference>
18        </ds:KeyInfo>
19    </ds:Signature>
20 </wsse:Security>
```

Example II

```
1  <S:Header>
2      <wsse:Security>
3          <xenc:ReferenceList>
4                  <xenc:DataReference URI="#bodyID"/>
5          </xenc:ReferenceList>
6    </wsse:Security>
7  </S:Header>
8  <S:Body>
9      <xenc:EncryptedData Id="bodyID">
10         <ds:KeyInfo>
11                 <ds:KeyName>CN=Hiroshi Maruyama, C=JP</ds:KeyName>
12         </ds:KeyInfo>
13         <xenc:CipherData>
14                 <xenc:CipherValue>...</xenc:CipherValue>
15         </xenc:CipherData>
16     </xenc:EncryptedData>
17 </S:Body>
```


Security Considerations

- ⌘ WS-Security is one of the building blocks for securing SOAP messages and intended to be used in conjunction with other security techniques.
- ⌘ Digital signatures alone do not provide message authentication.
  - → Beware of Replay Attacks (suggested usage of timestamps, message sequence numbers, message expiration, message correlation)
  - → When building trust into an application based on a digital signature there are other technologies, such as certificate evaluation, that must be incorporated
- ⌘ The combination of signing and encryption over a common data item may introduce some cryptographic vulnerability.
  - → Care should be taken by application designers not to introduce such vulnerabilities.

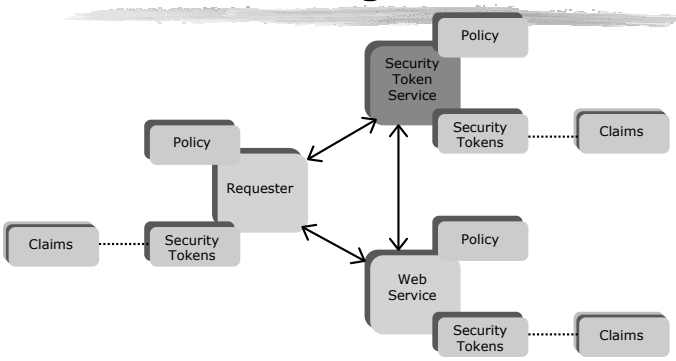
WS-Trust

- ✓ 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

Trust Model

- ✓ Messages MAY be required to prove a set of claims (e.g., name, key, permission, capability, etc.).
  - ✓ Messages without having the required proof of claims, SHOULD be ignored/rejected.
  - ✓ Requester MAY contact an appropriate authority (Security Token Service) which may require their own set of claims.
    - ✓ Security token services form the basis of trust.
  - ✓ A challenge response protocol MAY be required for freshness and proof-of-possession.

Trust Model Diagram



Request Example

- ⌘ Request Header
- ⌘ Token Type : defines the type of security token requested
- ⌘ Request Type: the action that has been requested
- ⌘ Base: references tokens that are used to validate the authenticity of a request
- ⌘ Supporting: references the supporting tokens used to authorize request

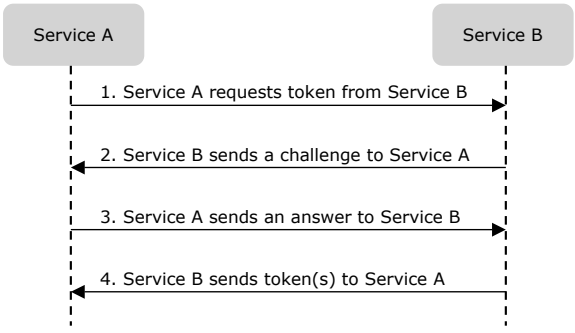
```
<RequestSecurityToken>
  <TokenType> … </TokenType>
  <RequestType>… </RequestType>
  <Base>…</Base>
  <Supporting>…</Supporting>
</RequestSecurityToken>
```


Response Example

- ⌘ Response Header
- ⌘ Defines the type of security token requested
- ⌘ Specifies the type of key used in the token
- ⌘ Specifies the size of the key returned
- ⌘ Specifies the scope to which this security token applies
- ⌘ Requested security token
- ⌘ Proof-of-possession token

```
<RequestSecurityTokenResponse>
    <TokenType>…</TokenType>
    <KeyType>…</KeyType>
    <KeySize>…</KeySize>
    <wsp:AppliesTo>…</wsp:AppliesTo>
    <RequestedSecurityToken>…
      </RequestedSecurityToken>
    <RequestedProofToken>…
      </RequestedProofToken>
</RequestSecurityTokenResponse>
```


Challenge Protocol



Trust Management

- **Fixed trust roots** – Simple fixed set of trust relationships between requestor and provider.
- **Trust hierarchies** – Builds on fixed trust roots but allows hierarchies of trust between requestor and provider.
- **Authentication service** – Essentially a fixed trust root where the provider only trusts the authentication service.

Trust Models

- **In-band** : The exchange request can be made either by a requestor or by another party on the requestor's behalf. If the security token service trusts the provided security token The basis of the trust is the relationship between the two security token services.
- **Out-of-Band** : An administrator or other trusted authority may designate that all tokens of a certain type are trusted.

Security Considerations

- All the security-sensitive message elements must be included in the scope of the message signature.
- Digital signatures alone do not provide message authentication.
  - →Beware of Replay Attacks (suggested usage of timestamps, message sequence numbers, message expiration, message correlation)
  - →When building trust into an application based on a digital signature there are other technologies, such as certificate evaluation, that must be incorporated
- Security token requests are susceptible to denial of service attacks.

WS-Policy

- ✓ Flexible and extensible XML-based syntax for expressing fine-grained access policies to Web Services
- ✓ Collections of policies could be composed
- ✓ Single policy grammar for heterogeneous requirements (e.g authentication, transport protpcol 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">
(4)                     MIIEZzCCA9CgAwIBAgIQEmtJZc0...
(5)       </wsse:BinarySecurityToken>
(6)       <wsse:BinarySecurityToken
                  ValueType="wsse:Kerberosv5TGT"
                  EncodingType="wsse:Base64Binary">
(7)                     JYTVjkvkjaOIJK76i7tuaeHJ...
(8)       </wsse:BinarySecurityToken>
(9)       <wsse:UsernameToken >
(10)            <wsse:Username>Claudio</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/XACML

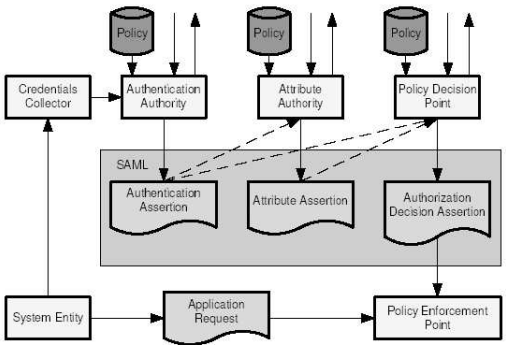
SAML rationale

- ⌘ SAML (*S*ecurity *A*ssertion *M*arkup *L*anguage): XML-based EXPRESSIVE and COMPLEX syntax for exchanging security data across heterogeneous, distributed systems boundaries
- ⌘ One major design goal for SAML is Single Sign-On (SSO)

SAML assertions

- ✓ 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 domain model



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 rationale

✓XACML (E*X*tensible *A*ccess *C*ontrol *M*arkup *L*anguage): XML-based syntax for expressing fine-grained access policies. Uniform language for heterogeneous, corporate-wide policies.

XACML (I)

- ✓ General-purpose authorization policy model and XML-based specification language
- ✓ XACML is independent of SAML specification
- ✓ Triple-based policy syntax: <Object, Subject, Action>
- ✓ Negative authorization supported

XACML (II)

- ✓ Input/output to the XACML policy processor defined as XACML context data structure
- ✓ Input data referred by XACML-specific attribute designator as well as XPath expression
- ✓ Extension points: function, identifier, data type, rule-combining algorithm, policy-combining algorithm, etc.
- ✓ A set of policies is combined by a higher level policy (policySet statement)

Rule 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.

Policies based on subject and resource attributes

- XACML provides facilities to support different characteristic of the subject :
  - → its identity.
  - → the subject's role [RBAC].
- XACML provides a standard way to reference the attributes defined in the LDAP series of specifications
  - → This is intended to encourage implementers to use standard attribute identifiers for some common subject attributes.

Policies 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 an 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

```
<Policy>
  <Target>
    <Resources>
    <Subjects>
    <Actions>
  <RuleSet ruleCombiningAlgId =
                        "DenyOverrides">
    <Rule ruleId="R1">
    <Rule ruleId="R2">
     …
    <Obligations>
  <RuleSet>
</Policy>
```

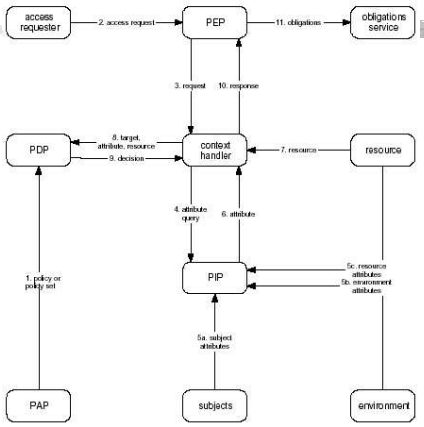
```
<Rule RuleId="R1"
      Effect="Permit">
  <Target>
    <Resources>
    <Subjects>
    <Actions>
  <Condition>
</Rule>
```

```
<Rule RuleId="R2"
      Effect="Deny">
  <Target>
    <Resources>
    <Subjects>
    <Actions>
  <Condition>
</Rule>
```


Architectural entities

- **Policy administration point (PAP)** - The system entity that creates a policy or policy set
- **Policy decision point (PDP)** - The system entity that evaluates applicable policy and renders an authorization decision
- **Policy enforcement point (PEP)** - The system entity that performs access control, by making decision requests and enforcing authorization decisions
- **Policy information point (PIP)** - The system entity that acts as a source of attribute values

Data flow diagram (I)



Data flow diagram (II)

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.
2. The access requester sends a request for access to the PEP.
3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource and action. The context handler constructs an XACML request context in accordance with steps 4,5,6 and 7.
4. Subject, resource and environment attributes may be requested from a PIP.
5. The PIP obtains the requested attributes.
6. The PIP returns the requested attributes to the context handler.
7. Optionally, the context handler includes the resource in the context.
8. The context handler sends a decision request, including the target, to the PDP. The PDP identifies the applicable policy and retrieves the required attributes and (optionally) the resource from the context handler. The PDP evaluates the policy.
9. The PDP returns the response context (including the authorization decision) to the context handler.
10. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
11. The PEP fulfills the obligations.
12. (Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

Policy Example

```
<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></Subjects>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
</Target> </Rule> </Policy>
```

XACML Request/Response Context Example (I)

```
<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>
```

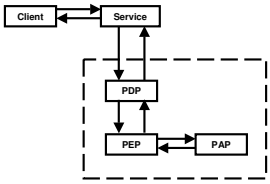

XACML Request/Response Context Example (II)

```
<Response>
   <Result>
      <Decision>Deny</Decision>
   </Result>
</Response>
```

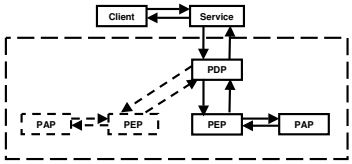

Architectural patterns for Web Service access control

Security Components

- ✓ PAP (Policy Administration Point)
  - ✓ Policies Repository
- ✓ PEP (Policy Evaluation Point)
  - ✓ Policies Enforcer
- ✓ PDP (Policy Decision Point)
  - ✓ Access Control Point



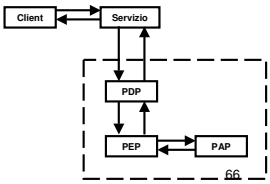
Main features



- ✓ Reusable and Extensible
- ✓ Modular and language-independent
- ✓ Suitable for high performance implementation
  - ✓ Components are themselves Web Services

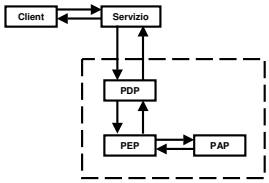
Policy Decision Point (PDP)

- ✓ Interface between a WS and the security infrastructure
- ✓ Outputs final access decision
- ✓ Instantiates one or multiple PEPs (e.g. if multiple parties can express an access policy)



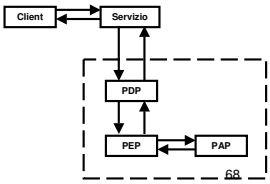
Policy Evaluation Point (PEP)

- ✓ Calls local PAP passing a request header and gets all policies applicable to that request
- ✓ Evaluates all applicable policies

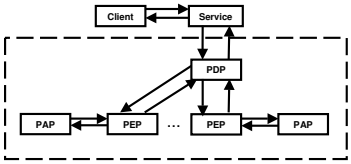


Policy Administration Point (PAP)

- ✓ A (possibly federated) policy repository
- ✓ Provides an administration WS interface for policies' insertion, deletion or modification
- ✓ Returns applicable policies

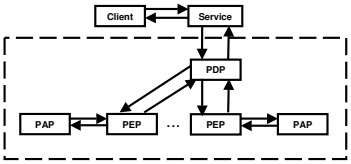


Multple PEPs



- ✓ PEPs correspond to the (possibly many) actors regulating access to a Ws
- ✓ Each PEP provides an answer
- ✓ Each PEP gets its policies from a separate PAP
- ✓ PDP aggregates the PEPs outcome in a final result

Chain authentication



- ⌘ Security components are themselves Web Services
- ⌘ All AC components may require caller authentication
- ⌘ Use the same AC infrastructure as "ordinary" WS

Semantics-aware AC

Three key aspects of knowledge representation

- ⌘ *Resource representation*. Writing access control policies where resources to be protected are pointed at via data identiers and access conditions are evaluated against their attribute values is not sufficient anymore. **It is important to be able to specify access control requirements about resources in terms of metadata describing them.**
- ⌘ *Context representation*. Distributed environments increase the amount of context information available at policy evaluation time (e.g., **location-based one**).
- ⌘ *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.

Extending XACML

- **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 forthe 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

```
<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/Image/chart457.png"/>
<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.

Evaluating a request

- Consider now a request to see presentation 7318.smi submitted by a user who presents to our system subject 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 dened in the domain ontologies, there is the subsumption: Instructor is a sub-class of Trainer".
- According to this subsumption, 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.

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

```
<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>
<SubjectAttributeDesignator
AttributeId="urn:ourdomain:attribute:metatag"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
</Subjects>
```

Resources

```
<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"/>
<rdf:predicate rdf:resource="http://ourdomain.it/MD/Schema/md-syntax#contains
    video"/>
<rdf:object rdf:nodeID="content"/>
</rdf:Statement>
<rdf:Statement rdf:nodeID="content">
<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>
```


Actions

```
<Actions>
<Action>
<AnyAction />
</Action>
</Actions>
```


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, to verify whether the requested resource satises 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.

Example revisited

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.

Problems

- Although our 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.

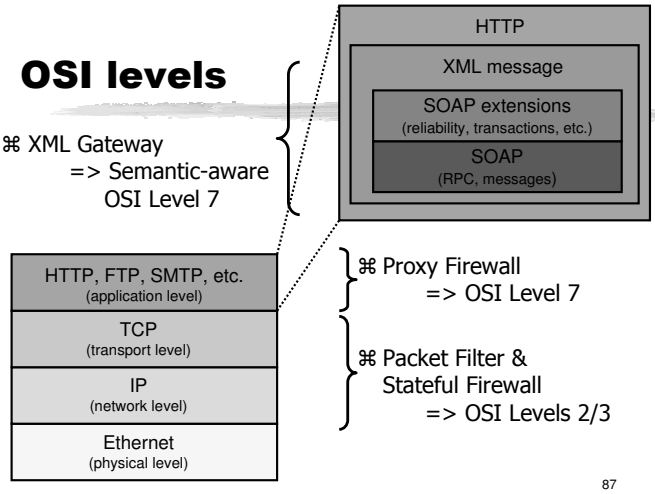
XML Gateway

Uncertain terminology

⌘ WS Security Gateway,
XML Gateway,
XML Firewall,
SOAP Proxy,
SOAP Firewall.

OSI levels

⌘ XML Gateway
=> Semantic-aware
OSI Level 7



⌘ Proxy Firewall
=> OSI Level 7

⌘ Packet Filter &
Stateful Firewall
=> OSI Levels 2/3

Features

⌘ Authorize service provision :

Is the SOAP request targeted to a service that the requester is authorized to access?

⌘ Content inspection :

Is the content of the SOAP request valid for the target service?

Architectural options

⌘Custom Coded Mode :

- ✓ Custom-build & platform-specific
- ✓ End-point security (web services)

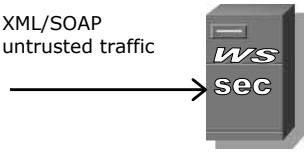
⌘Intercepting Agent Mode :

- ✓ Centralized XML/SOAP validation and access control
- ✓ End-point security (agents)
- ✓ Dependent on WS servers (agents)

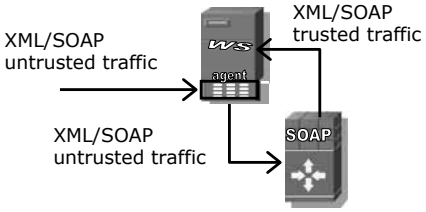
⌘ Gateway Mode :

- ✓ Centralized XML/SOAP validation and access control
- ✓ Perimetral security
- ✓ Independent from WS servers

Custom Coded Mode



Intercepting Agent Mode



Gateway Mode

XML Gateway: Rationale

- ⌘ Vendor Neutrality : To be independent from underlying systems (SOAP servers, WS implementation, etc.)
- ⌘ Perimetral Security :
  - ✓ single point of entry and single point of policy enforcement
  - ✓ enforce consistent and auditable security policies across multiple business systems
- ⌘ Performance : CPU-intensive tasks performed at the edge of the network and by specilized hosts

Examples of XML security threats

- ⌘ Sniffing : XML human-readable format
- ⌘ Tampering : illegal modification on data in transit
- ⌘ Buffer overflow : overriding of kernel data structure by unchecked input data
- ⌘ Denial-of-Service : traditional and XML level
- ⌘ Man-in-the-Middle : illegal impersonification
- ⌘ SQL injection : unchecked input data result in illegal SQL queries
- ⌘ Platform-specific vulnerabilities
- ⌘ XXE (Xml eXternal Entity) attack

Common Countermeasures

AUTHENTICATION

- ⌘ Checking username/passwords with LDAP directories
- ⌘ Verifying identities based on X.509 PKIs
- ⌘ Ckecking SAML assertions and attributes

CONFIDENTIALITY and INTEGRITY

- ⌘ Transport level encryption (SSL)
- ⌘ Message/element level encryption (XML Encription/WS-Security)
- ⌘ Message/element level digital signatures (XML Signature/WS-Security)

XXE (Xml eXternal Entity) attack (I)

XXE (Xml eXternal Entity) attack (II)

Overview:

- XXE (Xml eXternal Entity) attack is target on applications that parses XML input from untrusted sources using incorrectly configured XML parser.
- The application may be coerced to open arbitrary files and/or TCP connections.

Details:

- When an XML processor recognizes a reference to a parsed entity, in order to validate the document, the processor must include its replacement text.
- If the entity is external, and the processor is not attempting to validate the XML document, the processor may, but need not, include the entity's replacement text.

XXE (Xml eXternal Entity) attack (III)

SOAP and XMLRPC implementations accept XML encoded data over the network, sometimes from untrusted clients. They should be considered a prime targets for the XXE attack.

Products review:

Several SOAP and XMLRPC implementation were found vulnerable.

The following implementations were found NOT vulnerable :

Apache XML-RPC server, Marquée XML-RPC, WebLogic 6.1sp3 SOAP implementation, Phyton 2.2 SimpleXMLRPCserver, XMLRPC-J.

XXE (Xml eXternal Entity) attack (IV)

Successful exploitation may yield to:

- DoS on the parsing system
- TCP scans using HTTP external entities (including behind firewalls since application servers often have world view different from that of the attacker)
- Unauthorized access to data stored as XML files on the parsing system file system (of course the attacker still needs a way to get these data back)
- DoS on other systems (if parsing system is allowed to establish TCP connections to other systems)
- Doomsday scenario: A widely deployed and highly connected application vulnerable to this attack may be used for DDoS.

XML Gateway devices (I)

- Forum Sentry 1504 2.0. Forum Systems.
  `www.forumsys.com`
- DataPower XS40 XML Security Gateway 2.3. DataPower Technology.
  `www.datapower.com`
- Reactivity XML Firewall 2300. Reactivity.
  `www.reactivity.com`
- Westbridge XML Message Server. Westbridge Technology.
  `www.westbridgetech.com`
- Web Services Domain Boundary Controller 1.3. Xtradyne Technologies.
  `www.xtradyne.com`
- Trust Gateway 1.0.1. VeriSign.
  `www.verisign.com`

XML Gateway devices (II)

TESTS
(http://www.nwc.com/showitem.jhtml?docid=1421f2)

- Tools still not mature, standards still evolving
- No "ultimate security product"
- Loose WS-Security 1.0 standard support
- XML schema validation but weak SOAP semantic-aware content inspection
- Few generic-pattern matches for known SQL commands
- Not integrated with Web Application Firewalls
- Performance as a critical issue, some are based on hardware accelerators

XML Gateway devices (III)

EXPECTED FUNCTIONALITY

- Full XML content inspection
- Full WS-Security compliance
- Access control for both services and operations
- Integration with identity management systems as LDAP and Active Directory
- Zero-failure performance
- Schema validation and SOAP-envelope validation
- Logging and audit trails for transactions and policies

References

E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati. Extending policy languages to the semantic web. In Proc. of the International Conference on Web Engineering, Munich, Germany, July 2004.

E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in P2P systems. IEEE Transactions on Data and Knowledge Engineering, 15(4):840-854, July/August 2003.

E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Managing multiple and dependable identities. IEEE Internet Computing, November-December 2003.

E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati : A fine-grained access control system for XML documents. ACM Trans. Inf. Syst. Secur. 5(2): 169-202 (2002)

Many others:

Dipartimento di Tecnologie dell'Informazione (DTI) – Campus di Crema :
http://seclab.dti.unimi.it/