Seamlessly Securing Web Services Using Policies

Mifla Mashood, Gihan Wikramanayake

University of Colombo School of Computing

mifla17@gmail.com, gnw@ucsc.cmb.ac.lk
Abstract

• Web Services - A dominant force in distributed technology
• An undisputed STAR - Why?
  - *platform independency*
  - *ease of implementation*
  - *unprecedented support from major vendors*
  - *ability of seamlessly interfacing with legacy systems*
• Prying eyes !
• Building bridges across heterogeneous systems !!!
• Proposal - Cost effective seamless integration using policies
SSL vs. Message level security

- transport level
- a point-to-point protocol - exposed at each point
- encrypts the entire message
- no accountability, i.e. it is difficult to even prove that an SSL session existed

- message level
- an end-to-end protocol - not exposed at each point
- partial encryption of the message
- accountability
In order to successfully integrate with a nontrivial Web service, one must fully understand the service's XML contract along with any additional requirements, capabilities, and preferences (also referred to as policies). For example, just knowing that a service supports policies is not enough information to enable successful integration. The client needs to know in exact what sort of modifications are to be done on their part to communicate with the service. It also needs to know what security tokens it's capable of processing (such as UsernameToken, Kerberos tickets, or certificates), and which one it prefers. The client must also determine if the service requires signed messages. And if so, it must determine what token type must be used for the digital signatures.

Finally, the client must determine when to encrypt the messages, which algorithm to use, and how to exchange a shared key with the service.

Trying to integrate with a service without understanding these details is a stab in the dark. Without a standard way to convey policies, developers are left to convey them as they always have, i.e. through word-of-mouth and documentation.

WS-Policy which is a part of the well integrated architecture proposed by Microsoft for securing web services solves this problem by providing a standard way of conveying these policies.
Policy files at either end will manipulate SOAP messages in order to ensure security and other preferences at each end.

WS-Policy based components such as policy assertions, policy expressions, policy attachments etc... will be used for this purpose.
WS-Policy

WS-Policy provides a flexible and extensible grammar for expressing policies in a machine-readable XML format.

- **Policy Expression** - An XML representation of a policy

- **Policy Assertion** - It represents an individual preference, requirement, capability or other characteristic, and is the basic building block of a policy expression

```xml
<wsp:Policy xmlns:wsp="..." xmlns:wsu="..." wsu:Id="..." Name="..." TargetNamespace="...">
  <Assertion wsp:Usage="..." wsp:Preference="..." />
  <Assertion wsp:Usage="..." wsp:Preference="..." />
</wsp:Policy>
```
Integrating Security

Policy assertions allow you to specify the types of security tokens, signature formats, and encryption algorithms supported, required, or rejected by a given subject. Different preferences, usage levels etc... can be specified as depicted by the tables below.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsp:Required</td>
<td>The assertion must be applied to the subject. If the subject does not meet the criteria expressed in the assertion, a fault or error will occur.</td>
</tr>
<tr>
<td>wsp:Rejected</td>
<td>The assertion is explicitly not supported and, if present, will cause failure.</td>
</tr>
<tr>
<td>wsp:Optional</td>
<td>The assertion may be made of the subject, but is not required.</td>
</tr>
<tr>
<td>wsp:Ignored</td>
<td>The assertion is processed but ignored. That is, it can be specified, but no action will be taken as a result of it being specified. Subjects and requesters are informed that the policy will be ignored.</td>
</tr>
<tr>
<td>wsp:Observed</td>
<td>The assertion will be applied to all subjects, and requesters of the service are informed that the policy will be applied.</td>
</tr>
</tbody>
</table>

wsp:Usage Values

<table>
<thead>
<tr>
<th>Policy Assertion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:SecurityToken</td>
<td>Specifies a type of security token (defined by WS-Security)</td>
</tr>
<tr>
<td>wsse:Integrity</td>
<td>Specifies a signature format (defined by WS-Security)</td>
</tr>
<tr>
<td>wsse:Confidentiality</td>
<td>Specifies an encryption format (defined by WS-Security)</td>
</tr>
</tbody>
</table>
Conclusion

- Traditional approach
- Need for ubiquitous interoperability
- Sensitive data
- WSE 3.0 - The light at the end of the tunnel
- Channeling future work towards complete automation

References