Query Extension for SAML AuthnRequest (Draft)

Sampo Kellomäki, Symlabs, Inc.

April 22, 2008

Abstract

Proposal to pass more useful information in the AuthnRequest as required by real world deployment profiles.

Document History

02 22. April, 2008 Sampo

- Added background section

01 20. February 2008 Sampo

- Noted the general infeasibility of using AttributeConsumingService

00 12. February 2008 Sampo Kellomäki (sampo@symlabs.com)

- Proposal

1 Background

The author was engaged by the State Services Commission of the New Zealand Government to advise on the integration of SAML 2.0 into the [igovt] services offered by this government's Authentication Programme. A number of SAML-related issues arose, based on existing use cases and conceptual designs presented to me. I have taken those issues that I consider to have the greatest implications for the greatest number of real life deployments and proposed solutions for consideration by the SSTC. These are offered with the knowledge and support of the customer, who sought the views of their counterparts in other governments and concluded that there was wider interest in receiving SSTC guidance or standardisation efforts regarding these issues.

2 Introduction

<AuthnRequest>, defined in [SAML2core] conveys a very poor or constrained set of information from the SP to the IdP. Many real life deployments, or *deployment profiles*, have the following needs:

1. Convey version number of the *deployment profile*, distinct from SAML namespaces or the @Version attribute that describes the SAML specification version.

2. Dynamically express what attributes should be returned in the SSO transaction. This helps to promote minimal disclosure by not sending unnecessary attributes "just in case" as tends to happen in static configurations. It also provides operational convenience in configuring the systems.

   The metadata based approach of using existing @AttributeConsumingServiceIndex is inadequate as it is a static - configure time - mechanism, rather than dynamic runtime mechanism.

   <AttributeConsumingService> specification in the metadata seems rich enough, if only it were possible to enumerate a finite set of possible combinations of requested attributes. But such finite set may still be impractically large as it grows combinatorially.

   It seems more natural that if the number of combinations is large, one should be able to specify the requirements directly rather than use an index number. Explicit specification would be much less ambiguous and error prone than a hard to understand index that rigorously depends on having the right instance of metadata present.

3. Convey deployment dependent input to the authentication (and authorization) decision(s).

In general, the deployments need flexibility to define the data schema for such *input parameters* and are currently (2008) worried about interoperability of the COTS implementations in presence of such parameters and are not yet worried about interoperability across deployment domains. However, eventually interoperability across deployment domains will also be a concern and solution should be designed with foresight to address that future scenario as well.

As an immediate requirement, the deployments need some sort of container where they can safely pidgeon-hole all their customizations, with some guarantee that the pidgeon-hole will not break existing COTS software.

The existing SAML element `<AttributeQuery>` allows us (at least schemawise) to express both names of requested attributes as well as input parameters in form of attribute-value pairs (attribute-test in some interpretations, but that is compatible with the use of the `role` attribute, below).

Thus the problem really is how to include `<AttributeQuery>` in the same message as `<AuthnRequest>`. Note that doing two message exchanges, first `<AuthnRequest>` and then `<AttributeQuery>` is deemed inefficient and also inadequate because it would not allow input parameters to be supplied to the authentication (and authorization) process.

There are concerns that the aggregation of `<AuthnRequest>` and `<AttributeQuery>` is too bloated to be carried over redirect binding. Possible solutions are:

i. The deployment domain can restrict the attribute names and values to avoid bloat;

ii. The deployment domain can specify that some other binding, such as POST or artifact, is used to carry the `<AuthnRequest>`;

iii. We could try to change the XML culture to be less bloated (we would probably fail); or

iv. We could abandon the XML culture and roll our own, like [IDFF12] did for their redirect binding.

3 Proposal A: Extend <AuthnRequest> to have optional <AttributeQuery>

Example A

```
<sp:AuthnRequest
   xmlns:sp="urn:oasis:names:tc:SAML:2.1:protocol"
   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
   AssertionConsumerServiceIndex="0"
   ID="RNh43h2dqrtJLGvPCi2Cm"
   IssueInstant="2006-05-19T00:49:38Z"
   ProviderName="Symlabs demo SP 06"
   Version="2.0">
  <sa:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://cxp06.symlabs.com:7448/sp.xml</>
  <ds:Signature> ... </>
  <sp:NameIDPolicy
      AllowCreate="true"
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
  <sp:RequestedAuthnContext>
    <sa:AuthnContextClassRef>
      urn:nz:govt:authn:names:SAML:2.0:ac:ModStrength
    </></>
  <sp:AttributeQuery>
      <sa:Attribute Name="samsvers"><sa:AttributeValue>1.85</></>
      <sa:Attribute Name="cn"/>
      <sa:Attribute Name="o"/>
      <sa:Attribute Name="role"><sa:AttributeValue>director</></>
  </></>
```

This represents how SSTC perhaps should have defined the `<AuthnRequest>` in the first place.

Note how the `<AttributeQuery>` expresses the deployment profile version (`samsvers`) as an attribute-value pair. It also expresses the required attributes (`cn` and `o`) by naming them. Finally, it expresses an input parameter `role` as an attribute-value pair. The input parameter can also be interpretted as a *test* that the parameter must have the specified value.

This approach will break most schema-aware implementations. The SP implementations that only rely on XML well-formedness will continue to work (and hopefully pass the `<AttributeQuery>` to appropriate application layer).

An interesting property of this proposal is that it does not innovate any elements, but rather specifies a new composition of them. However, since the definition of `<AuthnRequest>` has changed, we need new namespace, e.g. `"urn:oasis:names:tc:SAML:2.1:protocol"`.

4 Proposal B: Use extension point to carry `<AttributeQuery>`

Example B

```
<sp:AuthnRequest
   xmlns:sp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
   AssertionConsumerServiceIndex="0"
   ID="RNh43h2dqrtJLGvPCi2Cm"
   IssueInstant="2006-05-19T00:49:38Z"
   ProviderName="Symlabs demo SP 06"
   Version="2.0">
 <sa:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
   https://cxp06.symlabs.com:7448/sp.xml</>
 <ds:Signature> ... </>
 <sp:Extensions>
   <sp:AttributeQuery>
     <sa:Attribute Name="samsvers"><sa:AttributeValue>1.85</></>
     <sa:Attribute Name="cn"/>
     <sa:Attribute Name="o"/>
     <sa:Attribute Name="role"><sa:AttributeValue>director</></>
 </></>
 <sp:NameIDPolicy
     AllowCreate="true"
     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
 <sp:RequestedAuthnContext>
   <sa:AuthnContextClassRef>
     urn:nz:govt:authn:names:SAML:2.0:ac:ModStrength
   </></></></>
```

This proposal has the advantage that the new material appears where extensions should appear. However, practical experience has raised some doubts about

whether schema-aware implementations really support the `<Extensions>` element in a meaningful way (or `xs:any` extension point in general).

Implementations relying only on well-formedness should not have any problem.

The `sp` namespace stays same as in original specs.

5 Proposal C: Define new element that carries <AuthnRequest> and <AttributeQuery>

Example C

```
<sp23:AuthnNAttrRequest
    xmlns:sp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:sp23="urn:oasis:names:tc:SAML:2.3:protocol">
  <ds:Signature> ... </>
  <sp:AuthnRequest
      AssertionConsumerServiceIndex="0"
      ID="RNh43h2dqrtJLGvPCi2Cm"
      IssueInstant="2006-05-19T00:49:38Z"
      ProviderName="Symlabs demo SP 06"
      Version="2.0">
    <sa:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
      https://cxp06.symlabs.com:7448/sp.xml</>
    <sp:NameIDPolicy
        AllowCreate="true"
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
    <sp:RequestedAuthnContext>
      <sa:AuthnContextClassRef>
        urn:nz:govt:authn:names:SAML:2.0:ac:ModStrength
      </></></></>
  <sp:AttributeQuery>
    <sa:Attribute Name="samsvers"><sa:AttributeValue>1.85</></>
    <sa:Attribute Name="cn"/>
    <sa:Attribute Name="o"/>
    <sa:Attribute Name="role"><sa:AttributeValue>director</></>
  </></>
```

This proposal keeps both `<AuthnRequest>` and `<AttributeQuery>` intact, but innovates the `<AuthnNAttrRequest>`, which of course necessitates a new namespace.

Question: should `<AuthnNAttrRequest>` carry the top level XML attributes like `@Version` and `@ID`? Or also some of the top level elements like `<Issuer>`.

This proposal formalizes the box-carring by creating a top-level element as specified by "best practises" advocated by some, but it seems it creates more problems than it solves. Apparently the SAML protocol request elements were not really designed to appear anywhere else than at top level.

6 Proposal D: Define new binding that allows box-carring <AuthnRequest> and <AttributeQuery>

For sake of illustration, we shall specify the input into the deflate-base64-URLencode layer of a hypothetical new `redir2` binding (the actual output being an inintelligeble base64 string):

Example D

```
encode(
        <sp:AuthnRequest
           xmlns:sp="urn:oasis:names:tc:SAML:2.0:protocol"
           xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
           AssertionConsumerServiceIndex="0"
           ID="RNh43h2dqrtJLGvPCi2Cm"
           IssueInstant="2006-05-19T00:49:38Z"
           ProviderName="Symlabs demo SP 06"
           Version="2.0">
         <sa:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
           https://cxp06.symlabs.com:7448/sp.xml</>
         <ds:Signature> ... </>
         <sp:NameIDPolicy
             AllowCreate="true"
             Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
         <sp:RequestedAuthnContext>
           <sa:AuthnContextClassRef>
             urn:nz:govt:authn:names:SAML:2.0:ac:ModStrength
           </></></></>
         <sp:AttributeQuery>
           <sa:Attribute Name="samsvers"><sa:AttributeValue>1.85</></>
           <sa:Attribute Name="cn"/>
           <sa:Attribute Name="o"/>
           <sa:Attribute Name="role"><sa:AttributeValue>director</></>
         </>
       )
```

Notes:

1. Solving the problem at the binding layer is simply wrong (although possible) approach!

2. There is no schema or namespace change.

3. Pattern of composition is not difficult to understand, although some "WS-I nitpicks" might complain about concatenating two messages.

4. Since this is new binding, no exisiting implementation is compatible

5. Implementing the new binding is easy for a programmer. Just add a loop where you used to process the unique (WS-I) request.

6. The identity context for the `<AttributeQuery>` would presumably be that which was established by `<AuthnRequest>`. This would need to be specified explicitly, i.e. out-of-order processing of the box-carred requests should be forbidden.

7. Same pattern works for both redirect and POST bindings.

7 Interrim Solution: Encode the Information as Query String in AuthenticationContextClassRef

The interrim solution is designed to break the least number of exisisting (as of 2007) SAML SP implementations. It does not use any schema level extension points and tries to introduce new functionality in the area that was already meant to be customizable. However, there is no knowing how limited the vendor implementations might be, so even this "solution" does not guarentee that there would not be breakage.

The main requirements placed on SP implementation are

1. Allow specification of multiple `<AuthnContextClassRef>` elements. The schema already allows this.

2. Allow, possibly dynamic, construction of at least one of the `<AuthnContextClassRef>` elements from the deployment parameters.

The fall back plan for "dumb" SPs is to only send the `<AuthnContextClassRef>` specifying the actual authentication level desired and determining the deployment profile out-of-band. This allows both "dumb" and "enlightned" SP implementations to reasonably coexist. IdP is assumed to understand both modes simultaneously.

Example I

```
<sp:AuthnRequest
   xmlns:sp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
   AssertionConsumerServiceIndex="0"
   ID="RNh43h2dqrtJLGvPCi2Cm"
   IssueInstant="2006-05-19T00:49:38Z"
   ProviderName="Symlabs demo SP 06"
   Version="2.0">
  <sa:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://cxp06.symlabs.com:7448/sp.xml</>
  <ds:Signature> ... </>
  <sp:NameIDPolicy
      AllowCreate="true"
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
  <sp:RequestedAuthnContext>
    <sa:AuthnContextClassRef>
      urn:nz:govt:authn:names:SAML:2.0:ac:ModStrength</>
    <sa:AuthnContextClassRef>
      http://registry.sams.ssc.govt.nz/AuthnParam
      ?samsvers=1.85\&ReqAttr=cn,o,role</>
  </></>
```

In this example, the second occurrance of the `<AuthnContextClassRef>` carries the deployment specific data. In particular, it contains a prefix that identifies the *deployment domain* and a Query String that contains the parameters defined by the domain, e.g. that `cn`, `o`, and `role` attributes are required this time.

`samsvers` reflects the requirement to define the version of the deployment domain specific profile.

`role` reflects the dynamic component as the SP may render the screen differently depending on whether `role` is unknown, insufficient, or adequate (e.g. read only wiki page vs. ability to edit).

The syntax of the Query String is basically up to the deployment domain and may be extended (e.g. "`ReqAttr=cn,role:director`", which would mean that `role` is required and must be "director").

The main consequence for COTS IdP software is that they need to be able to not crash upon seeing unforeseen `<AuthnContextClassRef>` and hopefully pass the

unforeseen values to appropriate layers that can interpret them. In many deployments, IdP can be customized (supporting this solution can be made a condition in procurement process), thus this should not be a major problem.

8 Author's Preference

Interrim solution (7) combined with the SSTC level (A) extension of `<AuthnRequest>`. While latter will require reimpementation by vendors, the reimplementation is fairly trivial. The namespace would naturally carry which version of the protocol is spoken.

9 Note on the Liberty ID Federation Framework (ID-FF) Guidance

Since Liberty ID Federation Framework [IDFF12] was the first Single Sign-On protocol to introduce the concept of `<AuthnRequest>`, and since the ID-FF variant suffers from the same short comings as SAML `<AuthnRequest>`, it would seem beneficial that the solution chosen above is also adopted for ID-FF, though this is a decision that Liberty Alliance has to make and publish.

Normative

[SAML2core] "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-core-2.0-os

[SAML2prof] "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-profiles-2.0-os

[SAML2bind] "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-bindings-2.0-os

[SAML2context] "Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-authn-context-2.0-os

[SAML2meta] Cantor, Moreh, Phipott, Maler, eds., "Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-metadata-2.0-os

[SAML2security] "Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-sec-consider-2.0-os

[SAML2conf] "Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-conformance-2.0-os

[SAML2glossary] "Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0", Oasis Standard, 15.3.2005, saml-glossary-2.0-os

[SAML11core] SAML 1.1 Core, OASIS, 2003

[SAML11bind] "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1", Oasis Standard, 2.9.2003, oasis-sstc-saml-bindings-1.1

[IDFF12] http://www.projectliberty.org/resources/specifications.php

[IDFF12meta] Peted Davis, Ed., "Liberty Metadata Description and Discovery Specification", version 1.1, Liberty Alliance Project, 2004. (liberty-metadata-v1.1.pdf)

[LibertyDisco] ID-WSF Discovery service 2.0

[RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.

[Schema1-2] Henry S. Thompson et al. (eds): XML Schema Part 1: Structures, 2nd Ed., WSC Recommendation, 28. Oct. 2004, http://www.w3.org/2002/XMLSchema

[XML] http://www.w3.org/TR/REC-xml

[XPATH] XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xpath.html

Informative

[igovt] http://www.e.govt.nz/services/authentication

[LibertyIDWSFOverview] Some ISF overview document