Compiled Sept. 22, 2003
Mary Thompson
Rebekah Lepro

Comments from mail:

The following comments are taken from the OGSA-AuthZ and SAML mailing list and are summarized here to provide background on some of the unresolved issues in the Attribute Requirements draft. We apologize for any errors in paraphrasing or taking things out of context.

1 Naming of attributes

1) It is hard to link the attribute namespace (namequalifier) and the attribute name when making policy statements without having a complex format for referencing attributes. Scott Cantor has experience with this problem in using SAML attributes in hibboleth. It is why XACML chose to have a one-part name.

If you want to distinguish "permisRole" from "BarcelonaRole" then define two different URIs for them. If you want to know they have the same format, then associate the same DataType with both of them.

Anne H. Anderson Email: Anne.Anderson@Sun.COM
Sun Microsystems Laboratories

2) Minutes from the SAML Face to Face meeting 8-10 Sept., 2003

SAML Work Item 21 Baseline attribute namespaces (W-21)
- X.500/LDAP attributes in SAML attribute assertions
  - many attributes defined, why reinvent
  - desiderata
    - deterministic mapping
    - works on all LDAP attributes
    - human readable attribute names
    - sensible use of SAML attribute schema
    - avoid registry (avoid BECOMING the registry)
    - Eve: can use DSML?

- X.500/LDAP attribute definition
  - X.500 attribute definition
    - position in type hierarchy (optional)
    - attribute syntax
    - equality, ordering, matching rules, cardinality, etc
    - OID
- SAML AttributeDesignator
  - AttributeNamespace, AttributeName
  - XML data types
  - no existing practice guidelines
  - Eve: value designed in extensibility
    - Rob: type part of the value, not part of the AttributeDesignator
  - Eve: AttributeDesignator designed for query; has no value
- XACML approach: XACML 1.0 appendix B.5
  - document URL naming
  - human-readable
  - not deterministic - many documents; probably need a registry
  - URL doesn't resolve
- OID approach
  - applies to all OIDs for all purposes
  - deterministic for all X.500/LDAP attributes
  - not human-readable; translation to local names required anyway?
    - Phil: numbers are better; semantics easier for numbers than names
    - Eve: not always...
    - Prateek intervenes....
- Short-name approach
  - all X.500/LDAP attributes have short-names
- Tony: why not use QNames?
  - Scott: QNames break digital signatures
- Eve: need to follow up on syntactic representation and vocabulary
- Scott: number 1 federation work item is to develop vocabulary
- Irving: many deployments have LDAP directories
- Scott: enable interop between such deployments by using LDAP attributes
- Frederick: does Liberty define in attributes about principals?
  - Scott: not in ID-FF.
- Eve: DSML example
  - XML vocabulary and process to represent LDAP directory info, query and update
  - A lot of products already talk DSML
- Eve: UBL (Universal Business Language) codelists
  - e.g. ISO currency and country codes
  - spec for codelist producers to map codelists to XML schemas
- Bob: deliverable convention for referring to X.500 attributes
- John Hughes: have to register own OIDs?
  - Bob: don't have to use this convention
  - Scott: goal not to introduce complexity to attribute naming
- Bob: relation to metadata
- John Linn: SAML should be neutral; smallest amount necessary
  - Bob: does that today
  - Prateek: nice to have standard syntax
- Bob: separate issue with attribute name syntax related to type?

*****

At the F2F I briefly described the UBL TC's design for "code lists", which have some similar properties to our attribute (and action) namespaces. We didn't consider SAML's precise current solution (simple name string + namespace URI) nor XACML's (one-part URI reference), but we did analyze a number of other solutions. I've attached a few slides that introduce the UBL solution (datatyped string with canonical metadata attributes, where the datatype is the interface between code list producers and consumers), and if anyone is interested I can also forward a long analysis document we did last year.

Eve

--

Eve Maler +1 781 442 3190
Sun Microsystems cell +1 781 354 9441
Web Products, Technologies, and Standards eve.maler @ sun.com

1.1 Wildcards

Some additional, historical discussions from the SAML and XACML list archives…
http://lists.oasis-open.org/archives/security-services/200201/msg00059.html
http://lists.oasis-open.org/archives/security-services/200202/msg00071.html
http://lists.oasis-open.org/archives/security-services/200105/msg00094.html
http://lists.oasis-open.org/archives/security-services/200108/msg00055.html
http://lists.oasis-open.org/archives/security-services/200106/msg00037.html

2 On attributes meta-data

2.1 Explicit types

1) Depend on schemas to define the type of the attribute. Each attribute type has its own schema, and the attribute value is an instance of that schema, identified by the schema namespace and element name. XrML takes this approach.

2) Depend on explicit DataType specifications, which may be XML schema namespace:element identifiers, but can also be identifiers assigned to particular generic or application-specific data types. XACML takes this approach.

Anne Anderson

For anyone who is interested, some XML-based systems do not include a DataType for what corresponds to an AttributeValue in XACML. Instead, the attribute value effectively is a "string" that is parsed as an XML schema instance. The schema-specified syntax is the specification of the data type of the attribute. There are good reasons not to use this approach in XACML or any system where attribute values must be manipulated, rather than simply passed around. An XACML PDP must know not only the syntax of an attribute value, but also the semantics for how to handle it in functions (compare it for greater or less than, add it to another value, etc.).

If attribute values were defined as schema instances, then not only would the PDP have to locate and process the schema associated with each attribute, but the PDP would also have to be augmented with code that understands the semantics of the schema-defined information. This means no "standard" language processor would be able to deal with attribute values in general. Some set of attribute value schemas and semantics could be defined as part of the standard, but the value of the "extensibility" gained would be questionable. The languages would be easily "extensible" in syntax, but not so easily extensible in semantics. It also means that companies could define proprietary schemas for AttributeValues, hindering interoperability.

XACML, by contrast, has chosen to define a rich set of DataTypes that are explicitly supported by all conforming PDP's. Most useful attributes can be defined in terms of these DataTypes and thus can supported by any standard XACML PDP. No new code modules are required except where a new DataType is defined.

XACML does support using an XML schema as the "DataType" for an attribute: use the schema namespace and element name as the URI of the DataType. This allows XACML to support Attributes that are defined using schemas as well as any other system. But the only operation on such DataTypes that could be standardized and supported in all PDPs is a byte-string comparison for equality between two instances of the AttributeValue (after canonicalization). It is probably just as easy to define the DataType for such a value as "string" and require operationally that the AttributeValue be the output of canonicalizing the schema instance.

--
Anne H. Anderson Email: Anne.Anderson@Sun.COM
Sun Microsystems Laboratories
1 Network Drive,UBUR02-311 Tel: 781/442-0928
Burlington, MA 01803-0902 USA Fax: 781/442-1692

3) Notes from presentation on Liberty ID-FF 1.2 spec and SAML, Scott Cantor at the SAML Face to Face meeting 8-10 Sept., 2003

Liberty conventions for saml:NameIdentifier
This is recent change, in 1.1 a saml:NameIdentifiers are just opaque strings
one-time (aka anonymous) IDs implying distinguishing format/use
"format" now used to express type of ID for Service Provider (SP).

"NameQualifier" is provider/affiliationID of SP to whom it's being provided, or with whom principle has affiliated.

LibertySubject has two nameIdentifiers, one "Liberty" and one SAML for the moment these are the same.

Encrypted nameIdentifier value meets ID-WSF case to give one SP an ID from another SP without compromising it
-indicated by specific Format attribute
-Liberty spec indicates how to avoid correlatable IDs

(Identity Provider, SP, nameIdentifier) is the triple that is unique, name by itself isn't unique

---

I agreed to send the SSTC my explanation for why using a schema
to define the data type of an attribute is not sufficient. But I
don't think that was ever an issue for the SSTC - it certainly is
not captured in the notes I made at the SAML F2F.

For anyone who is interested, some XML-based systems do not
include a DataType for what corresponds to an AttributeValue in
XACML. Instead, the attribute value effectively is a "string"
that is parsed as an XML schema instance. The schema-specified
syntax is the specification of the data type of the attribute.

There are good reasons not to use this approach in XACML or any
system where attribute values must be manipulated, rather than
simply passed around. An XACML PDP must know not only the syntax
of an attribute value, but also the semantics for how to handle
it in functions (compare it for greater or less than, add it to
another value, etc.).

If attribute values were defined as schema instances, then not
only would the PDP have to locate and process the schema
associated with each attribute, but the PDP would also have to be
augmented with code that understands the semantics of the
schema-defined information. This means no "standard" language
processor would be able to deal with attribute values in general.
Some set of attribute value schemas and semantics could be
defined as part of the standard, but the value of the
"extensibility" gained would be questionable. The languages
would be easily "extensible" in syntax, but not so easily
extensible in semantics. It also means that companies could
define proprietary schemas for AttributeValues, hindering
interoperability.

XACML, by contrast, has chosen to define a rich set of DataTypes
that are explicitly supported by all conforming PDP's. Most
useful attributes can be defined in terms of these DataTypes and
thus can supported by any standard XACML PDP. No new code
modules are required except where a new DataType is defined.

XACML does support using an XML schema as the "DataType" for an
attribute: use the schema namespace and element name as the URI
of the DataType. This allows XACML to support Attributes that
are defined using schemas as well as any other system. But the
only operation on such DataTypes that could be standardized and
supported in all PDPs is a byte-string comparison for equality
between two instances of the AttributeValue (after
canonicalization). It is probably just as easy to define the
DataType for such a value as "string" and require operationally
that the AttributeValue be the output of canonicalizing the
schema instance.

Anne
--

2.2 Null values

If the requester is not authorized to see an attribute, I think the attribute should just not be returned in the assertion. A "sniffer" still won't be able to tell the difference between an attribute that doesn't exist and an attribute they are unauthorized to see. Or did I miss something?

The main issue David is raising is: When I'm authorized to see an attribute, and that attribute really has a null/empty string value, it can't be sent to me in the assertion according to section 1.2.1.

You have dealt with this issue by defining the proprietary URN. The rules about defining proprietary URN's has always been a bit fuzzy to me, but I'm not sure that doing what you did conforms to URN registration "rules".
Perhaps someone better versed in URN namespace definitions can elaborate.

I think the "right" solution to this issue is to fix the SAML spec by permitting an AttributeValue element to be returned as an empty element -
i.e. <AttributeValue/>. One "might" say that this is already allowed by declaring that section 1.2.1 does not apply to the "user" data provided in this element.
Section 1.2.1 starts out with the sentence "All SAML string and URI reference values have the type xsd:String and xsd:anyURI, respectively...". So one could just say that the attribute value data is not a "SAML string". Thus it can be permissible to send an empty element for <AttributeValue>.

I'm not in favor of introducing a separate URN to distinguish the null/empty value.

Comments?

p.s. I've CC'ed the main TC list in case some folks there aren't on saml-dev and we need to possibly add this to a 1.1 errata and to the v2.0 edit list...

Rob Philpott RSA Security Inc. The Most Trusted Name in e-Security Tel: 781-515-7115 Mobile: 617-510-0893 Fax: 781-515-7020
mailto:rphilpott@rsasecurity.com

> -----Original Message----- From: Jim Christopher
> [mailto:jchristo@carolina.rr.com] Sent: Tuesday, September 02, 2003 5:41 PM
> To: saml-dev@lists.oasis-open.org Subject: Re: Attribute values or the lack
> therof
>
> Warren [et al.];
>
> We have the same situation. We used to leave the no-value attributes out of
> the response in order to prevent a requester from sniffing out semantics of
> our service they are not authorized to use. The drawback is that our service

> behaves the same way when a requester specifies an attribute the service
> doesn't recognize. That is, it's impossible to tell from our response whether
> a request contained an unrecognized attribute name or if the attribute value
> is "saml-null". Like all humans our service users make mistakes, so we opted
> to define a URN to represent the "saml-null" value (e.g.,
> urn:learningstation:names:entity-types:null) and instructed our service users
> to recognize that value.
>
> Doing this actually helped us solve several other issues in our SAML service.
> The service applies policy to every request based on the requester, the type
> of request being made (e.g., attribute, authorization decision, or
> authentication query), and the resource URN if applicable. We didn't have a
> way to specify the security policy for a resource-less attribute query vs. an
> attribute query with an empty resource, for instance, until we had a way to
> identify (internally) a "saml-null" value.
>
> HTH, jim christopher senior developer / r&d learningstation
>
> ----- Original Message ----- From: "Warren, David" <dwarren@rsasecurity.com>
> To: <saml-dev@lists.oasis-open.org> Sent: Tuesday, September 02, 2003
5:00 PM
> Subject: Attribute values or the lack therof
>
>
>> Hi fellow SAML'ers,
>>
>> How should an implementation send an empty value (i.e. like a NO-VALUE
value
>> in a database) for an attribute? The first idea I had was to send an
>> Attribute element with no AttributeValue but that seems to be explicitly
>> forbidden by the schema (no minOccurs attribute which means 1 is required, I
>> think). The second idea I had was to just specify an empty element (like
>> <AttributeValue/> or <AttributeValue></AttributeValue>) but section 1.2.1 of
>> the core spec (Assertions and Protocol, etc.) seems to disallow this.
>>
>> A similar problem comes up with trying to send an empty string (i.e. "").
>>
>> Have any other implementers solved this?
>>
>> David -- Obligatory .signatory David Warren   phone: 781-515-7152 RSA
>> Security Inc., 174 Middlesex Turnpike, Bedford, MA 01730
>> dwarren@rsasecurity.com

2.3 Attribute issuer

Subject :Re: [security-services] Request to Generalize Issuer - was
XACMLchange request
From :"Eve L. Maler" <eve.maler@sun.com>
To : security-services@lists.oasis-open.org
Date : Tue, 07 Jan 2003 11:17:39 -0500
I had an action (AI-25) to provide element-based and attribute-based
solutions to allow Issuer to carry NameQualifier and Format information.

Sorry for the truly horrible delay in doing this writeup.

You'll recall that we can't do this:

```
<attribute name="Issuer" type="saml:NameIdentifierType" use="required" />
```

because NameIdentifierType is a complex type, and attributes can't be bound to such.

An Element-Based Solution:
==========================

If we move Issuer information into an element structure, it's backwards-incompatible but is more consonant with our existing element-based NameIdentifier solution. We could either wait till SAML 2.0 to put this in, or make this new structure an optional feature in SAML 1.1 and let it sit alongside the existing Issuer-as-attribute information. However, note that Issuer is currently a required attribute and this can't change in SAML 1.1, so if people did use the new structure, they'd be duplicating some information in the instance.

Currently, Issuer information is provided like this:

```
<Assertion
  {other_assertion_metadata_attributes}
  Issuer=" http://www.example.com/AttribAuthority" ;>
  <Conditions>...</Conditions>
  <Advice>...</Advice>
  {assertion_content}
</Assertion>
```

We could either just add an <IssuerIdentifier> subelement, or we could add a more generic subelement that is prepared to hold future element-structured metadata that we dream up. Here I'll go with the former strategy, to keep it concrete and realistic.

The instance would now look like this (I'm keeping the old-style format string for now, to avoid confusing things further, but remember that we agreed to fix the fragment ID problem and invent new URNs):

```
<Assertion {metadata_attributes}>
  <IssuerIdentifier
  IssuerQualifier="www.example.com"
  Format=
  "urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName">
    AttribAuthority
  </IssuerIdentifier>
  <Conditions>...</Conditions>
  <Advice>...</Advice>
  {assertion_content}
</Assertion>
```

An Attribute-Based Solution:
============================

We can enhance the current Issuer attribute to allow for an IssuerQualifier and a Format to be provided in sister attributes. This is something that I think we can do in SAML 1.1, if we think the description of Issuer can tolerate the additional interpretations that we'd need to layer on top:

"The issuer of the assertion. The name of the issuer is provided as a string. The issuer name SHOULD be unambiguous to the intended relying parties. SAML authorities may use an identifier such as a URI reference that is designed to be unambiguous regardless of context."

However, in practice I think there is an interoperability problem because we've now got two fields to divide the old Issuer-field information into.

The instance would look like this (note that I broke up the original value into two places):

```
<Assertion
   {other_metadata_attributes}
   Issuer="AttribAuthority"
   IssuerQualifier="www.example.com"
   Format=
   "urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName">
   <Conditions>...</Conditions>
   <Advice>...</Advice>
   {assertion_content}
</Assertion>
```

Final Comments
==============

It may be that we want to keep the old Issuer attribute exactly as it is, semantics and all, and simply add alongside *whichever* solution (element-based or attribute-based) we decide is best for the future. It will require duplication of information in instances for anyone who wishes to get the benefits of articulated issuer info in SAML 1.1, but at least there's no tortured logic, and the transition to SAML 2.0 would be straightforward (drop the annoying old Issuer attribute and use the new solution exclusively).

I haven't sketched up the schema code yet; let's see which way we prefer to go first.

Eve

Hal Lockhart wrote:
> In the last meeting I agreed to provide specific changes required to

> allow the Issuer to contain NameQualifier and Foprmat, just as subject
> does, in order to provide more flexible matching of Issuer names. I also
> sugggested, without looking at the schema that the changes could be made
> backward compatable by using a Choice. However, it turns out that Issuer
> is an XML attribute.
>
> So it looks like the change required is to change the line:
>
>
>   <attribute name="Issuer" type="string" use="required" />
>
> to:
>
>
>   <attribute name="Issuer" type="saml:NameIdentifierType" use="required" />
>
>
> Since NameIdentifierType extends string and since NameQualifier and
> Format are use="optional" I think this is backward compatable, but I may
> be wrong.
>
> In the core spec, the simplest change would be to change the sentence on
> line 383 from:
>
> The name of the issuer is provided as a string.
>
> to:
>
> The name of the issuer is provided as a SAML NameIdentifier. The
> NameIdentifier is described in section 2.4.2.2.
>
> Alternatively, the description of NameIdentifier could be moved forward
> in the document.
>
> Hal
>

--
Eve Maler   +1 781 442 3190
Sun Microsystems   cell +1 781 354 9441
Web Technologies and Standards   eve.maler @ sun.com

[Date Prev] | [Thread Prev] | [Thread Next] | [Date Next] -- [Date Index] | [Thread Index] | [Elist Home]

3 Existing Attributes

3.1 VOMS

VOMS supports Groups (sub-groups later), Roles and Capabilities (a free text string to be interpreted by local sites for special processing).

The applications in EDG have identified use cases where groups *and* roles are both important. The users can choose the VO(s), group(s) and/or role(s) they request from the VOMS server.

Attributes from more than one VO is an important part of the model.

Dr David Kelsey
Particle Physics Department
Rutherford Appleton Laboratory
Chilton, DIDCOT, OX11 0QX, UK

3.2 Cardea

Currently, Cardea predominantly examines some of the LDAP attributes, e.g. organization, citizenship, location, etc. We also need to represent group and project membership and sponsorship. The other attributes represent whether or not specific documentation was verifiably signed (acceptable use statements, etc) by the requester.

Rebekah Lepro

PERMIS itself can use any LDAP attribute defined by anyone in its decision making. PERMIS applications typically define their own attributes. The only requirements placed on them are that the attribute type is identified by a globally unique object identifier (equivalent to namespace) and a string (these are LDAP requirements) and the attribute value is a simple string. New syntaxes could be supported for attribute values e.g. integers, by extensions to the PERMIS code.

Environmental parameters are different. Time constraints are built into PERMIS, as are string based constraints. Other types of constraints would need Java evaluating objects to be built for them.

David Chadwick

3.3 PERMIS

PERMIS is using privilege statements (XACML rules) as attributes that are shipped around. Format is an ASN.1 IA5String that holds the XACML encoded rule constructs. Given name is PrivilegeAttribute.

Markus Lorch

3.4 J2SE Policy Provider

* * *

A proposal for implementing a Java[TM] 2 Standard Edition (J2SE[TM]) Policy Provider that will accept XACML policies as the policy input can be found at

http://research.sun.com/projects/xacml/J2SEPolicyProvider.html

This would provide a way for Java applications using the standard
Java Policy API to use XACML policies and all the good stuff that
comes along with XACML:

- o dynamic policies,
- o use of arbitrary Subject and target attributes
  independant of the application,
- o distributed policies,
- o policies shared between multiple applications,
- o policies shared between Java and non-Java applications,
- o role based access control, and
- o resource labels.

3.5 SAML SSO via web browser

The document sstc-saml-metadata-2.0-draft-00.pdf has been submitted by Jahan Moreh (jmoreh@sigaba.com) to the OASIS Security Services TC document repository.

Document Description:
This document is a re-publication of SAML 1.1 Draft 07 Metadata specification. It defines metadata that describe the elements and attributes required to use the SAML Web Browser SSO Profiles. An MS Word version is also available.

Download Document:
http://www.oasis-open.org/apps/org/workgroup/security/download.php/3519/sstc-saml-metadata-2.0-draft-00.pdf

View Document Details:
http://www.oasis-open.org/apps/org/workgroup/security/document.php?document_id=3519

4 Conditions

4.1 Example an XACML condition:

```
    time  > 2003-09-10T13:20:00  and
    time < 2003-12-31T23:59:59
```

using XACML

```
<Condition
FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <!--  validity start -->
  <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-
greater-than"
      >
    <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-
one-and-only"
```

```
      >
      <EnvironmentAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:curre
nt-dateTime"
DataType="http://www.w3.org/2001/XMLSchema#dateTime" />
    </Apply>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#dateTime">2003-
09-10T13:20:00
-05:00</AttributeValue>
  </Apply>

 <!--  validity end -->
 <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-
less-than">
   <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-
one-and-only"
      >
     <EnvironmentAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:curre
nt-dateTime"
DataType="http://www.w3.org/2001/XMLSchema#dateTime" />
   </Apply>
   <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#dateTime">2003-
12-31T23:59:59
-05:00</AttributeValue>
 </Apply>
</Condition>
```

Markus Lorch

4.2 Should validity dates be optional or required?

I think it is important. For example in a university context, we want to assign the role of undergraduate to students for just one year, since this is their length of enrollment. Then we dont need to un-authorise them on mass at the end of the year, since their privileges will automatically time out. You can think of other similar scenarios e.g. privileges for team members of a 3 month project. You cant expect relying parties to know all this temporal information (e.g. access to a building, permission to print output etc.) therefore the assertion should contain its validity period

regards

David