CESNET Technical Report 5/2013

Standalone SAML Attribute Authority With Shibboleth

Ivan Novakov

Received 10. 12. 2013

Abstract

The article defines what a standalone attribute authority is and how it can be used to support inter-organizational activities in a federated authentication and authorization infrastructure. The article also provides a detailed technical description about the implementation of such attribute authority using Shibboleth IdP and how a Shibboleth SP can consume data from that authority.

Keywords: Shibboleth, attribute authority, SAML, identity federation

1 Introduction

1.1 What is a (standalone) attribute authority?

Attribute authority is generally a database containing users' information. This information is structured into user attributes and it is usually accessible through some kind of interface or protocol. Organizations usually have such kind of databases to store data about their users. They provide user information such as given name, family name, position, contact information etc.

However, there are cases, when some users' attributes are stored somewhere else. Then we are speaking of "standalone" attribute authority, e. g. run by a third party.

1.2 What are the use cases?

For example, when users from different organizations work together on a project, they usually have different responsibilities, they are divided into groups, they may have additional contact information etc. So there is additional information associated with the users. This information can be stored in a standalone database - an attribute authority.

Suppose there are multiple collaborative applications supporting the project (wikis, bug trackers etc.) and they all require to know user's identity as well as user's information related to the project (group membership, responsibilities). Instead of having each application store that information for itself, it is possible to fetch the identity from the user's home organization database and to get the project related information from the common attribute authority. The information is not duplicated and different applications don't need to worry about data synchronization.

So attribute authorities are suitable when you need to store common data for users from different organizations. An attribute authority can be implemented in various ways. Data may be stored in a SQL database or LDAP. Stored data may be accessible either natively or through an API (REST, SOAP). To be able to access the user's data, the requesting party should provide a unique user identifier, which is relevant for the particular attribute authority.

2 SAML Attribute Authority

SAML is a XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. The service provider may request for user attributes by sending an attribute query to the identity provider. The attribute query must contain user's unique identifier. It may optionally contain a list of requested attributes. If no attributes are specified, it indicates that all attributes are requested.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
     <samlp:AttributeQuery
         xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
         ID="_b09f0a158257dbeb920d5d734b59b87a"
         IssueInstant="2013-03-06T13:45:51Z" Version="2.0">
     <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
         https://sp.example.org/sp/shibboleth
       </saml:Issuer>
     <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
         <saml:NameID>john@home.org</saml:NameID>
       </saml:Subject>
     </samlp:AttributeQuery>
  </S:Body>
</S:Envelope>
```

The response is a standard SAML assertion, which contains an attribute statement:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
   <saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
        ID="_96f00291f71ed87ce7a4f2d94edd0410"
        InResponseTo="_b09f0a158257dbeb920d5d734b59b87a"
        IssueInstant="2013-03-06T13:45:53.722Z" Version="2.0">
      <saml2:Issuer
          xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
          Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
        https://idp.example.org/idp/shibboleth
```

```
</saml2:Issuer>
<saml2p:Status>
  <saml2p:StatusCode
      Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</saml2p:Status>
<saml2:Assertion
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="_70bc3017a67f78529a97a93522252a81"
    IssueInstant="2013-03-06T13:45:53.722Z" Version="2.0">
  <saml2:Issuer
     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://idp.example.org/idp/shibboleth
  </saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID>john@home.org</saml2:NameID>
    <saml2:SubjectConfirmation
        Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches">
      <saml2:SubjectConfirmationData
          Address="2001:718:1:e::233:205"
          InResponseTo="_b09f0a158257dbeb920d5d734b59b87a"
          NotOnOrAfter="2013-03-06T13:50:53.722Z"/>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions
      NotBefore="2013-03-06T13:45:53.722Z"
      NotOnOrAfter="2013-03-06T13:50:53.722Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>
        https://sp.example.org/sp/shibboleth
      </saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AttributeStatement>
    <saml2:Attribute FriendlyName="o"
        Name="urn:oid:2.5.4.10"
        NameFormat="urn:oasis:names:tc:SAML:2.0:
                    attrname-format:uri">
      <saml2:AttributeValue
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="xs:string">
        Project Inc.
      </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
        FriendlyName="eduPersonScopedAffiliation"
```

```
            Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9"
            NameFormat="urn:oasis:names:tc:SAML:2.0:
                        attrname-format:uri">
          <saml2:AttributeValue
              xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="xs:string">
            member@project.org
          </saml2:AttributeValue>
        </saml2:Attribute>
      </saml2:AttributeStatement>
    </saml2:Assertion>
  </saml2p:Response>
 </soap11:Body>
</soap11:Envelope>
```

Having a SAML attribute authority is particularly useful if you have a SAML based identity federation. A service provider may be configured to retrieve attributes from different attribute authorities. Attributes are exchanged on the middleware level and the underlying applications just consume them in the usual way. There is no need to modify the applications to be able to use those additional attributes.

SAML also provides a trust framework allowing the service providers to talk to attribute authorities in the same way as they do with identity providers, so no other explicit authentication method is required to be implemented.

3 Attribute authority based on Shibboleth IdP

3.1 Shibboleth IdP specifics

The role of the identity provider is to authenticate the user and issue the corresponding authentication statement to the service provider. Additionally, an attribute statement containing a set of user attributes is included in the response. So the identity provider acts also as an attribute authority.

Authentication and attribute resolution are processed separately - first, the user is being authenticated and then the acquired user identifier is used for attribute retrieval. So if we omit the authentication part, we can use a Shibboleth IdP instance as a standalone attribute authority.

In such scenario, the service provider first redirects the user to the identity provider of his home organization. The user authenticates there and then he is redirected back to the service provider with a successful authentication statement and a set of attributes from his home organization.

After obtaining the attributes the service provider initiates an offline attribute request to the standalone attribute authority. Offline here means that the service provider contacts the attribute authority directly, not through the user's browser as during the session initiation. A user identifier must be included in the attribute

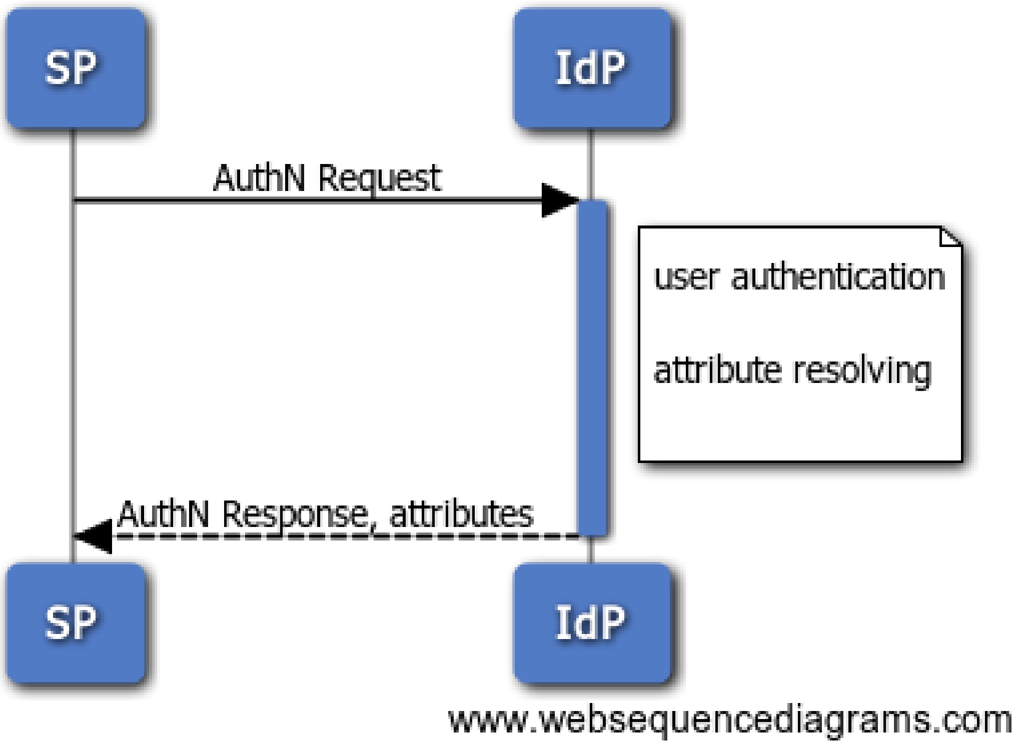

Figure 1. Attribute exchange between SP and IdP

request. Such identifier is usually based on attributes received from other sources - usually from the home organization.

The attribute authority returns a response, which contains a set of user attributes. The service provider then merges those attributes with the attributes obtained earlier.

3.2 Configuration

When configuring a Shibboleth IdP instance as an attribute authority we proceed as usual. We don't have to configure authentication, but we still need to set the right entity ID, the relevant metadata sources and key/certificate pair in the `relying-party.xml`file.

Next, we need to configure how attributes are resolved and what release policies are applied. Again, we proceed as if configuring a standard Shibboleth IdP. In `attribute-resolver.xml`we define data connectors with the corresponding attributes and in `attribute-filter.xml`we specify the release policies.

There is one small but significant difference though - the way the user is identified. In a standard Shibboleth IdP instance, after the user has been authenticated and his identity verified, a transient identifier is being issued. The transient identifier is an opaque short-time string identifier coupled with the user's identity. Its lifetime is usually very short (5 minutes). The transient identifier is used by the attribute resolver to get the identity of the user and fetch his attributes. The transient identifier is translated back to the user's identity through a so called transient principal connector which is defined in the `attribute-resolver.xml`file.

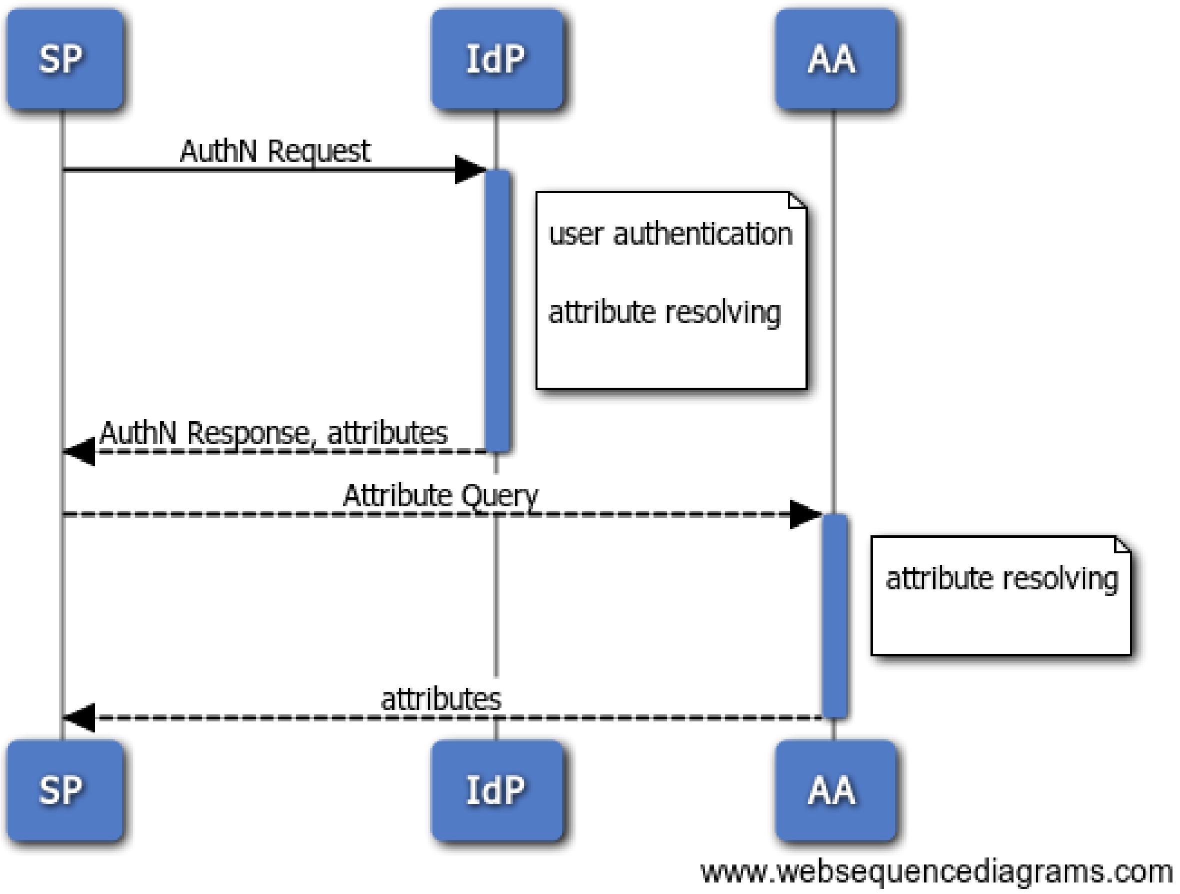

Figure 1. Attribute exchange between SP, IdP and AA

But in a standalone attribute authority the user's identity is not available internally, it is received through the attribute query. Instead of the transient principal connector, which maps the transient ID to the user's identity, we need to use a direct principal connector, which performs no mapping and just uses the passed identifier as the user's identity.

In the attribute-resolver.xml file we need to comment out the transient principal connector, which handles the `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` identifier format:

```
<!--
<resolver:PrincipalConnector xsi:type="pc:Transient"
   id="saml1Unspec"
  nameIDFormat="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"/>
-->
```

And then we define the direct principal connector:

```
<resolver:PrincipalConnector xsi:type="pc:Direct"
   id="saml2Direct"
  nameIDFormat="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
/>
```

4 Configuring the service provider

Usually, the service provider receives user's attributes during the single sign-on (SSO) process. The identity provider "pushes" the attributes together with the authentication response. However, Shibboleth SP can pull attributes from additional sources through the attribute resolver plugin. It can be configured through the *AttributeResolver*element in the main configuration file `shibboleth2.xml`.

The default configuration contains one *AttributeResolver*element with type *Query*. It means, that if no attributes have been received during the SSO, an attribute query should be issued back to the identity provider.

In order to fetch attributes from a standalone attribute authority we need to define an *AttributeResolver*of the type *SimpleAggregation*:

```
<AttributeResolver type="SimpleAggregation"
    attributeId="eppn"
    format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
  <Entity>https://aa.example.org/aa/shibboleth</Entity>
</AttributeResolver>
```

In this basic example we specify that the "eppn" attribute has to be used as a user identifier when querying the authority. The *Entity*sub-element contains the entity ID of the attribute authority. It is possible to define multiple *Entity*elements. It is also possible to fetch the entity ID from a previously resolved attribute using the *EntityReference*sub-element. That attribute can have multiple values resulting in multiple attribute queries per each value. Both approaches (*Entity*and *EntityReference*) can be combined and the attribute queries are issued in the order the elements are defined:

```
<AttributeResolver type="SimpleAggregation"
    attributeId="eppn"
    format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
  <Entity>https://aa.example.org/aa/shibboleth</Entity>
  <EntityReference>external-aa</EntityReference>
</AttributeResolver>
```

If the attribute authority is not a part of the federation of the service provider or for some reason its metadata are not included in the metadata sources defined in the application configuration, it is possible to define a dedicated metadata source using the *MetadataProvider*sub-element:

```
<AttributeResolver type="SimpleAggregation"
    attributeId="eppn"
    format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
  <Entity>https://aa.example.org/aa/shibboleth</Entity>
  <MetadataProvider type="XML"
    uri="http://aa.example.org/metadata/aa-metadata.xml"
    backingFilePath="aa-metadata.xml"
    reloadInterval="3600">
  </MetadataProvider>
```

```
</AttributeResolver>
```

It is also possible to use an alternative attribute extractors and filters than the ones defined on the application level:

```
<AttributeResolver type="SimpleAggregation"
    attributeId="eppn"
    format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
  <Entity>https://aa.example.org/aa/shibboleth</Entity>
  <AttributeExtractor type="XML" validate="true"
      reloadChanges="false" path="aa-attribute-map.xml"/>
  <AttributeFilter type="XML" validate="true"
      path="aa-attribute-policy.xml"/>
</AttributeResolver>
```

You can find detailed description of all the configuration options in the original documentation.

5 Privacy considerations

The attribute requests issued by the service provider to the attribute authority are done without any user interaction. Normally, they are issued during the session initiation process. But the service provider can send the request at any other time. While this may be useful if the service provider needs to keep locally stored user data current, it also allows a misbehaving service provider to mine user data from the attribute authority.

One possible solution is to control access to the attribute authority through custom metadata. Instead of making the authority available to the whole federation, choose only the service providers, which really need it and put them in a custom metadata file consumed by the authority.

If for some reason the attribute authority has to be a part of the federation, the attribute release can be controlled by setting appropriate rules in the attribute release policies. These rules may define, which service providers can receive attributes and which attributes exactly they are allowed to consume. It is always a good practice to release the minimum needed information.

6 Conclusion

An attribute authority can be used to store and provide common attributes for users from different organizations. Different interfaces and protocols may be used to access the user data. SAML attribute authorities are particularly useful when there is an existing SAML identity federation with established policies and trust. Moreover, the user's attributes are retrieved during the single sign-on process and merged with the other attributes, so the applications receive them in a standard way and there is no need to modify them to make them consume those extra attributes. Currently, Shibboleth Identity Provider can be used as a standalone attribute authority out of the box and Shibboleth Service Provider can be easily configured to issue attribute queries.

References

[1] http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[2] https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language

[3] https://wiki.shibboleth.net/confluence/display/SHIB2/NameIDAttributes

[4] https://wiki.shibboleth.net/confluence/display/SHIB2/
DirectPrincipalConnector

[5] https://wiki.shibboleth.net/confluence/display/SHIB2/
NativeSPAttributeResolver

[6] https://rnd.feide.no/2009/08/24/sp-centric_attribute_aggregation/