Risoluzione e filtraggio degli attributi con Shibboleth

v. 0.1

7 Maggio 2011

Revisioni

Versione	Data	Descrizione	Note
0.1	07/05/2011	Versione iniziale	Raffaele Conte[1]

1 Istituto di Fisiologia Clinica, CNR, Pisa <raffaele.conte@cnr.it>

La gestione degli attributi del profilo utente, in Shibboleth, comporta due principali fasi: il recupero o la definizione dei valori relativi ad un attributo (risoluzione) e il filtraggio degli stessi in funzione di altri valori o dell'interlocutore.

Risoluzione degli attributi

Shibboleth 2.x offre diversi strumenti per recuperare attributi da sorgenti esterne, manipolarli o eventualmente definirli al proprio interno. Di seguito verranno mostrate alcune configurazioni per recuperare gli attributi richiesti da IDEM, valide nella maggior parte dei casi. Per configurazioni particolari si rimanda alla documentazione di Shibboleth 2.x [SHIBATTR].

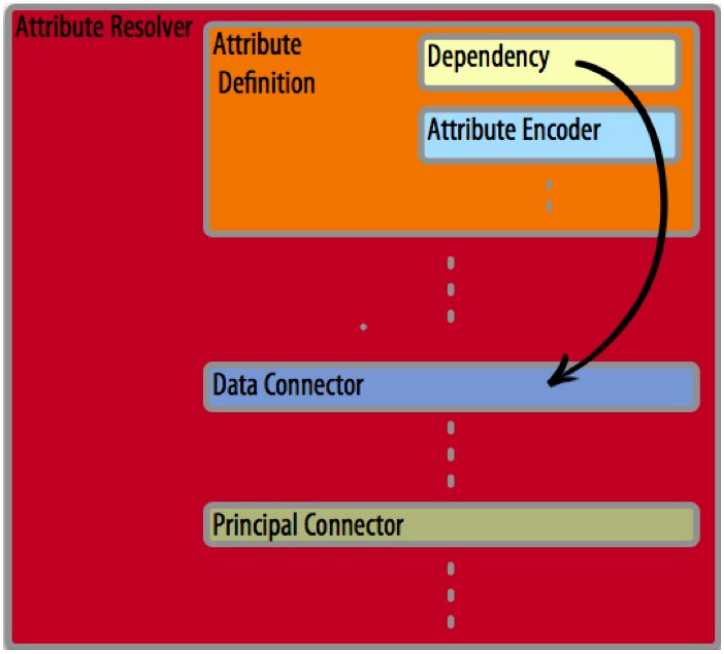

Figura 1: Attribute Resolver

Il file di configurazione in cui indicare le definizioni descritte di seguito e con cui Shibboleth esegue la risoluzione degli attributi è [SHIB HOME]/conf/attribute-resolver.xml (dove [SHIB HOME] rappresenta la directory di installazione di Shibboleth IdP).

Le definizioni sono indicate in XML, pertanto ogni definizione ha un tag di apertura ed uno di chiusura. La struttura del file può essere rappresentata come in figura 1.

L'**Attribute Resolver** è quindi il sottosistema di Shibboleth responsabile del recupero, l'eventuale modifica e la codifica degli attributi.

All'interno dell'attribute resolver si trovano:

- l'**Attribute Definition:** un plugin che crea un singolo attributo tramite trasformazione di altri attributi o informazioni. Shibboleth supporta come *attribute definition: simple, scoping, regex, mapping, template, scripting, principal name e principal authentication method.*
- il **Data Connector:** un plugin che crea più attributi staticamente o da sorgenti esterne come LDAP e DB. Shibboleth supporta come data connectors: *static, LDAP,* relational database, computed *ID e stored ID.*

È importante notare che il Data Connector da solo non è sufficiente per comunicare informazioni alla controparte: solo gli attributi che passano per un'*attribute definition* escono dal resolver.

Ai fini della configurazione degli attributi definiti da IDEM è possibile ignorare il **Principal Connector** (particolare costrutto per la definizione di un'attributo). È sufficiente sapere che è utilizzato per comunicare il *transientID*, definito all'interno dello stesso IdP, tramite cui le asserzioni sono associate ad uno specifico soggetto. Queste definizioni (più di una per compatibilità con

versioni precedenti di Shibboleth) sono già definite all'interno dell'attribute-resolver.xml e non è necessario modificarle.

All'interno di un'attribute definition troviamo:

- la **Dependency** (una sola) che lega la *definition* al *connector*;
- **l'Attribute Encoder**: un plugin per convertire un attributo in un formato specifico per un protocollo, es. un attributo SAML. È possibile cha all'interno di una definition siano presenti più codifiche (in Shibboleth 2.x ne compaiono almeno due per compatibilità con le precedenti versioni). Il SP utilizzerà quella a lui "comprensibile"

Definizione di un attributo di tipo “static”: eduPersonOrgDN

In alcuni casi il valore di un attributo non varia in funzione del tempo né dell'utente a cui è riferito e si può considerare quindi statico. Un attributo di questo tipo potrebbe essere eduPersonOrgDN, il cui valore, nella maggior parte dei casi, è riferito all'organizzazione rappresentata dall'IdP che comunica l'attributo e non varia per nessuno degli utenti gestiti.

In questo caso l'attributo può essere definito innanzitutto creandolo tramite un data connector;

```
<resolver:DataConnector id="staticAttributes" xsi:type="Static"
      xmlns="urn:mace:shibboleth:2.0:resolver:dc">
   <Attribute id="eduPersonOrgDN">
      <Value>o=Istituto di Fisiologia Clinica,o=CNR</Value>
   </Attribute>
</resolver:DataConnector>
```

Nell'esempio, al *data connector* è stato assegnato arbitrariamente un'identificativo, *staticAttributes*, unico per ogni *data connector*. Il plurale utilizzato nel nome fa ben intendere che in effetti è sufficiente un data connector per definire un insieme di attributi dello stesso tipo (o da una stessa sorgente). In questo caso il *data connector* è di tipo statico (`xsi:type="Static"`). Si intuisce che esistono diversi tipi di *data connector* e ogni tipo ha un set diverso di parametri (si faccia riferimento alla documentazione di Shibboleth per maggiori dettagli [SHIBATTR]).

All'interno del *data connector* sono inizializzati gli attributi ed i relativi valori. In questo caso il solo *eduPersonOrgDN*. Anche in questo caso l'identificativo è indicato arbitrariamente utilizzando il nome più logico. Il nome utilizzato è tuttavia totalmente indipendente da quello con cui l'attributo viene comunicato al relying party.

Come detto in precedenza non è sufficiente creare o recuperare da una fonte esterna un attributo per comunicarlo al Service Provider ma è necessario *definirlo* tramite un *attribute definition*:

```
<resolver:AttributeDefinition id="eduPersonOrgDN" xsi:type="Simple"
            xmlns="urn:mace:shibboleth:2.0:resolver:ad"
            sourceAttributeID="eduPersonOrgDN">
   <resolver:Dependency ref="staticAttributes" />
   <resolver:AttributeEncoder xsi:type="SAML1String"
            xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:mace:dir:attribute-def:eduPersonOrgDN" />
```

```
   <resolver:AttributeEncoder xsi:type="SAML2String"
            xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.1.1.3"
            friendlyName="eduPersonOrgDN" />
</resolver:AttributeDefinition>
```

La definizione appena vista sostanzialmente dice che l'attributo che si va a definire, identificato come **eduPersonOrgDN** (`id="eduPersonOrgDN"` è l'identificativo dell'attribute definition), fa riferimento all'attributo eduPersonOrgDN (`sourceAttributeID="eduPersonOrgDN"`) creato all'interno del *data connector* **staticAttributes** (`resolver:Dependency ref="staticAttributes"`).

L'attributo viene presentato al SP con il nome urn:mace:dir:attribute-def:eduPersonOrgDN (`name="urn:mace:dir:attribute-def:eduPersonOrgDN"`), nel caso di un SP Shibboleth 1.x, o come urn:oid:1.3.6.1.4.1.5923.1.1.1.3 (`name="urn:oid:1.3.6.1.4.1.5923.1.1.1.3" friendlyName="eduPersonOrgDN"`) nel caso di SP di tipo Shibboleth 2.x (o comunque di SP aderenti alle specifiche di SAML 2).

Come per il *data connector* anche *l'attribute definition* può essere di tipo diverso in funzione della specifica necessità. Nel caso appena descritto è stato utilizzato il tipo *simple* (`xsi:type="Simple"`).

È necessario notare che:

- la dipendenza (*dependency*) deve essere dichiarata prima di ogni altro parametro di configurazione;
- ogni attribute definition ha un unico id;
- ogni tipo di *attribute definition* ha un diverso insieme di parametri.

Un “semplice” attributo ricavato da LDAP: cn

La maggior parte degli attributi descrittivi del profilo di un utente, possono essere ricavati da una fonte esterna, tipicamente un server LDAP o un DataBase relazionale.

In questi casi la definizione dell'attributo può essere effettuata con il tipo simple come nel caso precedente. Supponiamo ad esempio di voler definire l'attributo **cn** contenente il **common name** dell'utente. L'attribute definition sarà quindi:

```
<resolver:AttributeDefinition id="cn" xsi:type="Simple"
      xmlns="urn:mace:shibboleth:2.0:resolver:ad" sourceAttributeID="cn">
   <resolver:Dependency ref="myLDAP" />
   <resolver:AttributeEncoder ... />
</resolver:AttributeDefinition>
```

Tralasciando la definizione dell'encoder, simile a quanto visto in precedenza, va notato solo che la definizione fa riferimento ad un attributo (`sourceAttributeID="cn"`) definito in un *data connector* identificato come *myLDAP*. Il *data connector* in questione è quindi:

```
<resolver:DataConnector id="myLDAP" xsi:type="LDAPDirectory"
      xmlns="urn:mace:shibboleth:2.0:resolver:dc"
      ldapURL="ldap://ldap.example.org" baseDN="ou=people,dc=example,dc=org"
```

```
      principal="uid=myservice,ou=system"
      principalCredential="myServicePassword">
   <FilterTemplate>
       <![CDATA[(uid=$requestContext.principalName)]]>
   </FilterTemplate>
</resolver:DataConnector>
```

Il data connector appena indicato è definito come myLDAP ed è di tipo *LDAPDirectory*. Per questo motivo richiede i parametri, il cui significato è evidente, *ldapURL, baseDN, principal* e *principalCredential* tramite cui accedere al server LDAP contenente le informazioni.

Si noti che la definizione del server LDAP appena descritta ha lo scopo di recuperare gli attributi relativi al profilo utente ed è distinta da quella indicata nel file relying-party.xml necessaria per l'autenticazione dell'utente stesso.

Con le definizioni appena descritte è possibile recuperare e trasmettere la maggior parte degli attributi indicati dalla Federazione. Definizioni particolari possono essere necessarie per **eduPersonScopedAffiliation** e **eduPersonTargetedID** per le quali si rimanda rispettivamente alle appendici B e C.

Filtraggio degli attributi

Una volta definiti gli attributi è necessario definire le regole tramite cui gli stessi vengono rilasciati. Il file in cui queste regole sono specificate, anche in questo caso in formato XML, è [SHIB HOME]/conf/attribute-filter.xml.

Il file può essere rappresentato come in figura 2.

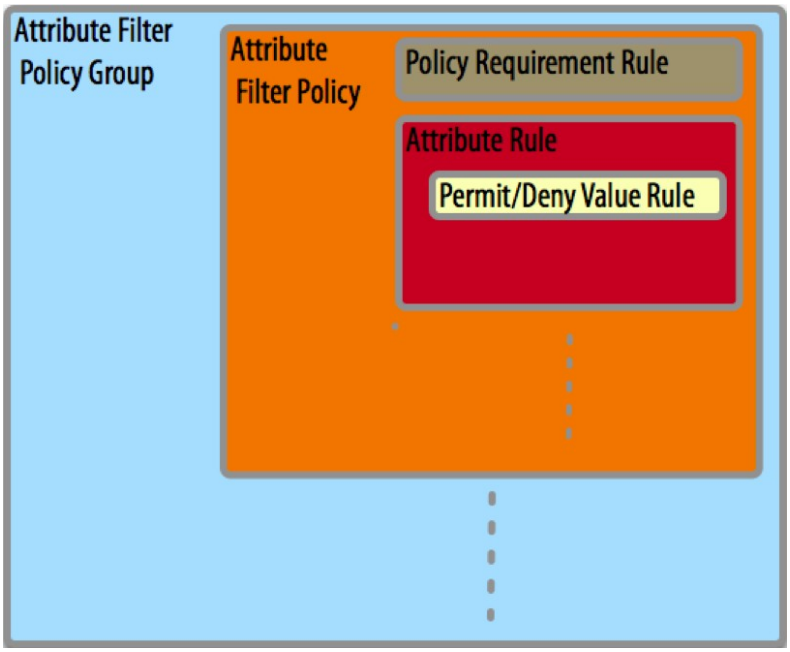

Figura 2: Attribute Filter

Il file è quindi composto dai seguenti elementi:

- **Attribute Filter Policy Group:** è l'elemento root per il plugin di filtraggio. Posso no esisterne più di uno (ragionevolmente in file distinti) ognuno con un unico id.

- **Attribute Filter Policy:** è l'elemento che definisce una policy. Possono esisterne più di uno all'interno del gruppo, per ognuna delle diverse policy. Ognuno di essi contiene un trigger, che determina se la policy deve essere applicata ad una specifi ca richiesta, ed un insieme di filtri sugli attributi ed i rispettivi valori.

- **Policy Requirement Rule:** Ne deve esistere una e solo una per ogni Attribute Fil

ter Policy. Determina se la specifica policy deve essere applicata ad un determinato richiedente.

- **Attribute Rule:** è una regola specifica per un singolo attributo. Possono esisterne più di una per i diversi attributi.
- **Permit/Deny Value Rule:** determina se lo specifico valore dell'attributo può esse re rilasciato al relying party.

Rilasciare un attributo a tutti: transientID

La policy più semplice è ovviamente quella che consente di rilasciare qualsiasi valore (`PermitValueRule xsi:type="basic:ANY"`) per un particolare attributo a chiunque si presenti (`PolicyRequirementRule xsi:type="basic:ANY"`). Un attributo di questo tipo è certamente il *transientId* (definito in precedenza con un *principal connector*) e la policy, il cui identificativo (univoco) è *releaseTransientIdToAnyone*, è la seguente:

```
<AttributeFilterPolicy id="releaseTransientIdToAnyone">
   <PolicyRequirementRule xsi:type="basic:ANY" />
   <AttributeRule attributeID="transientId">
       <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
</AttributeFilterPolicy>
```

Ovviamente esistono diversi tipi di *PolicyRequirementRule* e di *PermitValueRule*/*DenyValueRule* ognuna con uno specifico set di parametri.

Rilasciare un attributo solo ad alcuni SP

Volendo rilasciare alcuni attributi solo ad un particolare SP è possibile utilizzare una policy di tipo *AttributeRequesterString* come di seguito:

```
<AttributeFilterPolicy id="releaseToSpExampleOrg">
   <PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
      value="http://sp.example.org"/>
   <AttributeRule attributeID="email">
       <PermitValueRule xsi:type="basic:ANY" />
   </AttributeRule>
</AttributeFilterPolicy>
```

In questo caso tutti i valori (`PermitValueRule xsi:type="basic:ANY"`) dell'attributo **email** (nell'esempio vi è un solo attributo ma potrebbero essere di più con più *AttributeRule* all'interno della policy) sono rilasciati solo se l'entity ID del relying party è http://sp.example.org.

Negare un attributo in funzione di un altro

L'ultimo esempio, la cui applicazione potrebbe risultare utile in diverse occasioni, riguarda il caso in cui si voglia negare un attributo, magari in funzione di un valore contenuto in un'altro attributo.

```
<AttributeFilterPolicy id="denyOnPrivacyAttr">
   <PolicyRequirementRule xsi:type="basic:AttributeValueString"
      attributeID="privacyAttr" value="true" />
   <AttributeRule attributeID="firstName">
      <DenyValueRule xsi:type="basic:ANY" />
   </AttributeRule>
   <AttributeRule attributeID="surname">
      <DenyValueRule xsi:type="basic:ANY" />
   </AttributeRule>
</AttributeFilterPolicy>
```

Il tipo di policy utilizzata è *l'AttributeValueString* che richiede i parametri *attributeID*, l'attributo da valutare, ed il valore (*value*) in funzione del quale applicare la policy. La policy si applica se *privacyAttr=true* agli attributi *firstName* e *surname* (`AttributeRule attributeID="firstName"` e `AttributeRule attributeID="surname"`) negando (DenyValueRule) qualsiasi valore di questi (`DenyValueRule xsi:type="basic:ANY"`).

Le possibilità di filtraggio sono diverse e si rimanda alla documentazione di Shibboleth per ulteriori casi [SHIBFILTER].

Bibliografia

Shibboleth

[SHIB] Shibboleth

http://shibboleth.internet2.edu/

[SHIBATTR] Define and Release a New Attribute in an IdP

https://spaces.internet2.edu/display/SHIB2/IdPAddAttribute

[SHIBFILT] Define a New Attribute Filter

https://spaces.internet2.edu/display/SHIB2/IdPAddAttributeFilter