Invitation based authorization in federated systems

Jaime Pérez
Middleware Engineer
1. The authorization problem in federated systems

- “Not every user that can authenticate in a federation should be allowed to access everywhere”
2. Traditional authorization methods

- Filtering users by means of their **unique attributes**, i.e. name, e-mail, a targeted ID or a personal code.

  uid: jaime
  name: jaime
  mail: jaime.perez@rediris.es
  eduPersonTargetedID: 123456789
  shacPersonalUniqueCode: c0f57bdb21e2ffa58fd56eebc8a93d9b
Pros

• Flexible, may allow any kind of attribute.

Cons

• Service Providers need to know what exact attributes the users will send and with which values on a per user basis.
• Difficult to manage for SPs.
• Breaks easily if any attribute changes.
• Forces to know a lot about users beforehand.
2. Traditional authorization methods

• Filtering users by means of a specific attribute:

  eduPersonEntitlement:
  urn:mace:rediris.es:sir:wiki:admin
Pros

• Easy to deploy for Service Providers, just expect one attribute/value pair for authorized users.

Cons

• Not very flexible. Uses just a specific attribute.
• Difficult to manage for Identity Providers, as they need to add attributes for every user entitled to access any resource.
• Service Providers don’t have control on whom access their resources. They can change the rules but since they are well-known, any IdP can misbehave and gain unauthorized access until SPs react.
2. Traditional authorization methods

- Filtering users by means of an attribute authority:
Pros

• Service Providers may gain control over their resources.

• Avoids Identity Providers to worry about authorization.

Cons

• Although you store specific attributes for authorized users, you still need to identify them and therefore know some of their attributes in advance.

• Equivalent to the attribute filtering, though you can group attributes in a common place for different resources.
2. Traditional authorization methods

- Filtering users by means of an external authorization engine:
Pros
• Service Providers don’t need to worry about user’s attributes.
• Avoids Identity Providers to worry about authorization.

Cons
• Who runs the authorization engine?
• Which rules govern authorization?
• Is authorization trusted for Service Providers?
• Is authorization really transparent for Identity Providers, or the authorization engine lacks the same problems already seen?
3. Our specific use case: the SIR federation

- We run a federation with hundreds of Service Providers and dozens of Identity Providers.
- The federation looks like a hub. Both SPs and IdPs connect directly to an exchange point where data is collected and served adequately.
- Extremely flexible and relaxed, so you (as a SP) can’t expect anything coming from a user except a few, mandatory attributes.
- We need to provide a way for Service Providers to authorize users in the federation without knowing anything about them beforehand.
The SIR architecture

One Ring to bring them all and in the darkness bind them
In the Land of Mordor where the Shadows lie.
SIR mandatory attributes

SIRdemo: entorno de pruebas del SIR

Si ha accedido a esta página significa que su proveedor de identidad (IdP) ha autenticado correctamente al usuario.

El objetivo de esta aplicación es comprobar que su IdP está enviando los atributos requeridos, tal como se especifica en la guía para la solicitud de inclusión de IdP en el SIR.

Comprobación de atributos

La siguiente lista son los atributos requeridos por el SIR:

<table>
<thead>
<tr>
<th>Atributo</th>
<th>Valor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePTI</td>
<td>ee09145c38727ae8542ce7a30a20e4ec</td>
</tr>
<tr>
<td>ePA</td>
<td>staff</td>
</tr>
<tr>
<td>sHO</td>
<td>rediris.es</td>
</tr>
</tbody>
</table>

¡Enhorabuena! Su IdP está enviando todos los atributos que se recomiendan para participar en el SIR. Póngase en contacto con el equipo técnico del SIR para solicitar la inclusión de su IdP en el entorno de producción.

Información enviada por el IdP

A continuación se muestran los datos enviados por su proveedor de identidad. En el caso de que esté teniendo problemas a la hora de emitir los atributos solicitados en la guía, no dude en ponerse en contacto con el equipo técnico del SIR incluyendo la información mostrada más abajo:

- **Assertion**: sPUC=urn:mace:terena.org:shac:personalUniqueCode:es:rediris:sir:mbid: (md5)6ae694c934523b083404765255798c, cn=Jaime Perez Crespo, mail=jaime.perez@rediris.es, uid=jaime_ePTI=ee09145c38727ae8542ce7a30a20e4ec, ePA=staff, sHO=rediris.es, ePE=urn:mace:rediris.es:entitlement:wiki:jra5
- **Issued by**: newbabela
- **Valid until**: 2018-10-10
What do we need?

- A way to control the access to our resources, the finest-grained as possible, preferably on a per-user basis.
- No need to ask Identity Providers to set specific attributes for their users. It’s painful for users, for IdP administrators and for us.
- Easy to manage.
- Compatible with group management tools.
- Compatible with older solutions (attribute authorities or even authorization engines).
- Easy to use for users and for ourselves.
4. Our solution

1. Send invite
2. Register invite #1234567
3. Invite #1234567
4. Authentication + #1234567
5. Authn
6. Reference #1234567
7. Authz

Admin

Authorization engine

Authorized users

Service Provider

User

SIR
Using e-mail to trigger federated logins

• Our proposal makes use of e-mail to trigger a login for the user we want to authorize, so that we can look and record his attributes for future reference.

• Invitation e-mails are targeted and one-time.

• Once the user logs into the application, his attributes are recorded and will be used to track him down; the invitation is then removed and the user authorized.
5. Security considerations

- Multiple users shouldn’t be allowed by following the same invitation, therefore invitations are one-time. Once a user follows them, they are completely removed.
- Since invitations are targeted, only the appropriate user should be able to use them. We solve this issue by requiring an opaque, unique identifier (shacPersonalUniqueCode) computed by means of the institutional e-mail address and comparing to the address the invite was sent to.
6. Implementation

- The implementation of the system described here is written in PHP, as an independent authorization engine part of our phpPoA software, which allows applications to check user authentication and authorization with a couple of method calls.

- There are improvements contributed by our institutions to provide more functionality, i.e. an administrative interface/group management tool, allowing authorization and invites to expire and much more.
Current status

- The system has been deployed in multiple applications inside SIR.
- It has been integrated with mailing lists, so that all subscribers of a list are entitled to access to a specific resource just because they are members of the list. When they are removed, their authorization is removed too.
- There are plans to attach it to an attribute authority/authorization engine, so that it can be provided as a service inside the federation that could be used by SPs to manage access by their own.
Thanks for listening

Questions?