GÉANT OIDC plugin for Shibboleth IdP

Henri Mikkonen & Janne Lauros

CSC – IT Center for Science Ltd.

30th NORDUnet conference 2018

18-20 September 2018

Elsinore, Denmark

Overview

- Background and motivation
- Project principles
- Highlights so far
- Current status
- RP configuration example
- Upcoming highlights

Background

- Shibboleth is very widely used in the R&E identity federations worldwide
  - In many federations Shibboleth IdP is used almost in every IdP (e.g. Haka in Finland, SWITCH-AAI in Switzerland)
  - It remains popular even though many commercial and OS implementations exist
- OIDC is more popular as e.g. social media providers use OAuth2 based protocols
  - However, it won't replace SAML quickly
  - Both protocols need to be supported by the federations IdPs
  - Ideally without additional software and with similar configuration logic / structure
    - The protocols resemble each other
    - Deployments and configurations can be quite complicated (multi-factor authentication, IDS integrations, etc..)

Motivation

- GÉANT 4-2 JRA3 T3 has many activities related to OIDC
  - The OIDC federation spec (see Roland's presentation)
  - Python reference implementation (RP and OP)
  - RP library extensions for various programming languages implementing the spec
- In November 2016 workshop in Finland, the idea for this development project was born
  - Developers: Janne Lauros and Henri Mikkonen from CSC
    - Prior experience on Shibboleth extensions: MPASSid, Haka MFA, …
  - Collaborate with the Shibboleth consortium from the beginning

Project Principles

- Implement the OIDC support as Shibboleth IdP plugin
  - Should be possible to install the plugin to an existing (SAML) deployment
  - Aim at implementing as orthodox plugin as possible
- Exploit the protocol-independent features of Shibboleth IdP
  - Authentication engine (incl. MFA), attribute engine, session management, relying party configuration, consent, interceptors, etc..
- Collaborate actively with the Shibboleth development team
  - Aim at doing the implementation as they would (if they had time)

The Project Highlights so far (1/2)

- November 2016: Agreed to propose the plugin development for GÉANT and Shibboleth consortium
- March 2017: Presentation of the initial technical plans to the Shibboleth team
  - Use of Nimbus library as the OIDC message-level implementation
  - Implement the implicit flow first, as it resembles saml2int with attribute push
- April 2017: Started the implementation process
  - https://github.com/CSCfi/shibboleth-idp-oidc-extension
  - Vagrant configuration + Ansible playbook for easy provisioning of VMs

The Project Highlights so far (2/2)

- December 2017: The first alpha release (v0.5.0a)
  - Implicit flow
  - Open dynamic registration
- March 2018: The second alpha release (v0.6.0a)
  - Authorization code and hybrid flows
  - UserInfo-endpoint
- June 2018: The third alpha release (v0.7.0a)
  - General improvements, e.g. related to clustering
  - Added minor features

Current Status

- Mostly compliant with all the OIDC OP conformance profiles v3.0
  - https://openid.net/wordpress-content/uploads/2018/06/OpenID-Connect-Conformance-Profiles.pdf
  - Currently only open dynamic registration (2.1.5) - i.e. no RP authentication
- After v0.7.0a, the underlying Shibboleth IdP codebase changed to 3.4-SNAPSHOT
  - The 3.4 codebase offers new features that simplifies our implementation
- Some successful testing deployments reports, but more still needed
  - Documentation needs improvements though
  - We still provide only Vagrant + Ansible, not good way to install on top of existing deployment

RP configuration example

How to setup trusted RPs?

- There’s no directly matching standard to SAML metadata in OIDC
  - Dynamic client registration spec defines client metadata
- Trusted RPs can be configured statically via filesystem
  - one-by-one per file, or multiple RPs in single file(s) by JSON array
    -
      {
      "scope":"openid info profile email address phone",
      "redirect_uris":["https://rp.example.org/authz_cb"],
      "client_id":"demo_rp",
      "response_types":["id_token"]
      }

- Dynamic registration exploits *storage service* interface, so for instance in-memory or RDBMS can be used for storing the data

Enable OIDC profiles in the relying-party.xml

```
…
<bean id="shibboleth.DefaultRelyingParty" p:responderIdLookupStrategy-ref="profileResponderIdLookupFunction"
parent="RelyingParty">
   <property name="profileConfigurations">
      <list>
         <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
         <ref bean="SAML2.ECP" />
         <ref bean="SAML2.Logout" />
         <ref bean="SAML2.AttributeQuery" />
         <ref bean="SAML2.ArtifactResolution" />
         <bean parent="OIDC.SSO" p:postAuthenticationFlows="attribute-release" />
         <bean parent="OIDC.UserInfo"/>
      </list>
   </property>
</bean>
…
```

Configure OIDC attribute names in attribute-resolver.xml

```
…
   <AttributeDefinition id="email" xsi:type="Template">
      <Dependency ref="uid" />
      <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-def:mail" encodeType="false" />
      <AttributeEncoder xsi:type="SAML2String" name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail"
         encodeType="false" />
      <AttributeEncoder xsi:type="oidcext:OIDCString" name="email" />
      <Template><![CDATA[
            ${uid}@example.org
        ]]></Template>
      <SourceAttribute>uid</SourceAttribute>
   </AttributeDefinition>
…
```

Configure OIDC attribute filtering in attribute-filter.xml

```
...
   <AttributeFilterPolicy id="OPENID_SCOPE_EMAIL">
      <PolicyRequirementRule xsi:type="oidcext:OIDCScope" value="email" />
      <AttributeRule attributeID="email">
         <PermitValueRule xsi:type="ANY" />
      </AttributeRule>
      <AttributeRule attributeID="email_verified">
         <PermitValueRule xsi:type="ANY" />
      </AttributeRule>
   </AttributeFilterPolicy>
...
```

Upcoming Highlights

- Hands-on tutorials
  - CSC office in Espoo, Finland (October 2018)
  - Technology Exchange 2018 in Orlando, Florida, U.S. (October 2018)
  - GÉANT office in Amsterdam, Netherlands (December 2018)
- Beta release soon after Shibboleth IdP 3.4 (estimated October 2018)
  - Installable plug-in instead of Ansible playbook
- First official release before end of the year
  - Maintenance, support and further development proposed to continue in GÉANT 4-3 WP5

Thank you

Any questions?