deutsches forschungsnetz

OpenID Connect am Shibboleth IdP

DFN-AAI Workshop

Silke Meyer (smeyer@dfn.de)

Überblick

- Plugins
- Konfiguration
- Discovery und Client Registrierung
- unsere Dokumentation

Showcase

- 2 OpenID Connect Provider:
  - idp1.local, manuelle Client-Registrierung (`/opt/shibboleth-idp/idp1.local`)
  - idp2.local, dynamische Client-Reg. (`/opt/shibboleth-idp/idp2.local`)
- 2 Relying Parties: Apache-Modul mod_auth_oidc
  - sp1.local → registriert an idp1 (`/etc/apache2/sites-enabled/sp1.conf`)
  - sp2.local → dyn. Reg. an idp2 (`/etc/apache2/sites-enabled/sp2.conf`)

IdP-Plugins

Plugins und Module im Shibboleth IDP

- zusätzliche Funktionalitäten für den IdP, inkl. einheitlicher Routinen zur Wartung (`./bin/module.sh`)
- Aktivierung von Modulen = Hinterlegen spezifischer Konfigurationsdateien im Dateibaum
- bla.idpnew: Konfigs, die bei Update neu hinterlegt wurden
- bla.idpsave: Konfigs, die bei Deinstallation gesichert wurden

Drei Plugins für den Shibboleth IDP als OP

- OIDC Common und OIDC Config (beide Voraussetzung)
  - keine eigenen Features, sondern „Meta-Plugins“ für OAuth- und OIDC Java-Komponenten
  - verhindern doppelte Bibliotheken und andere mögliche Konflikte
- OIDC OP
  - die eigentliche OpenID Connect Provider-Funktionalität
  - eigene Konfigurationsdateien und Skripte

Aktivierung des OP-Moduls

```
▶ root@ubuntu:/opt/shibboleth-idp/idp1.local# ./bin/module.sh -e idp.oidc.OP
  Enabling idp.oidc.OP…
          conf/oidc.properties created                                  → allgemeine Einstellungen
          conf/oidc-clientinfo-resolvers.xml created                    → woher Client-Informationen geholt werden sollen
          conf/oidc-credentials.xml created
          conf/attributes/oidc-claim-rules.xml created                  → OIDC Claims für die Attribute Registry
          conf/authn/oauth2client-authn-config.xml created
          conf/examples/oidc-attribute-filter.xml created
          conf/examples/oidc-attribute-resolver.xml created
          static/openid-configuration.json created                      → generische OP-JSON-Metadaten
          bin/jwtgen.sh created
          bin/jwtgen.bat created
          bin/lib/json-web-key-generator-0.8.2-jar-with-dependencies.jar created
          bin/issue-access-token.sh created
          bin/issue-access-token.bat created
          bin/oidc-clients.sh created
          bin/oidc-clients.bat created
  [OK]
```

Konfiguration

OP-Metadaten bereitstellen

- JSON-Metadaten unter standardisierter Request URI zugänglich machen:

  `/.well-known/openid-configuration`

- Vorlage static/openid-configuration.json
- Redirect in Webserver-Konfiguration
- Inhalt der Metadaten u.a.:
  - Identifier des OP („issuer“)
  - Kommunikations-URLs (Authorization-, Token- und UserInfo-Endpunkte)
  - unterstützte Flows und Algorithmen
  - unterstützte Identifier, Scopes und Claims

conf/oidc.properties I

- Issuer ID (vergleichbar mit SAML Entity ID)
- Storage der JSON Web Key Sets anderer Parteien
- Authentifizierungsmethoden am Token Endpunkt
- Gültigkeitsdauer der Tokens
- Signing Keys für Tokens, zus. Verschlüsselung ja/nein
- Consent Storage des IdP verwenden oder Consent Information in Tokens schreiben?

conf/oidc.properties II

- Quellattribut, Algo, Salt für Generierung von Sub Claims (ähnl. persistent ID)
- Claim Handling Rules:
  - Attribute, die immer im Access Token hinterlegt werden sollen
  - Attribute, die immer im ID Token hinterlegt werden sollen
  - Attribute, die nie in der UserInfo Response enthalten sein sollen
- Vertrauen über SAML-Metadaten herstellen? ja/nein
- mögliche Umschaltung auf OAuth 2.0 Authorization

Profile für OIDC-Support aktivieren

- `conf/relying-party.xml`
- offen zugängliche Profile (**`UnverifiedRelyingParty`**):
  - OIDC.Configuration
  - Key Advertisement: OIDC.Keyset
- für registrierte RPs zugängliche Profile (**`DefaultRelyingParty`**):
  - OIDC.SSO
  - OIDC.UserInfo
  - OAUTH2.Revocation: Endpunkt für Revocation Requests (Mitteilung von Clients, dass Access bzw. Refresh Tokens nicht mehr benötigt werden (RFC 7009 OAuth 2.0 Token Revocation)).
  - OAUTH2.Introspection: Abfragemethode, mit der Resource Server Metadaten über Tokens beim Auth. Server anfragen können (RFC 7662 OAuth 2.0 Token Introspection)

Claims

- Konfiguration analog zu SAML-Attributen, gleiches Prozedere zum Auflösen und Filtern (siehe AAI-Wiki)
- `conf/attribute-resolver.xml` und `conf/attribute-filter.xml`
- speziell: der **„Sub" Claim**, *der* Unique Identifier für den jeweiligen Resource Owner, ein OP muss einen Sub Claim liefern („public" oder „pairwise", siehe (OPSubClaim))
- semantisch vergleichbar mit SAML Subject Id (non-targeted) und Pairwise Id (targeted), gleiche Datenquellen sind empfohlen

Claims

- ```
  <AttributeDefinition id="subject-pairwise" xsi:type="Simple"
  activationConditionRef="shibboleth.oidc.Conditions.PairwiseRequired">
    <InputDataConnector ref="StoredId" attributeNames="persistentID" />
  </AttributeDefinition>
  ```
- Activation Condition: ist wahr, wenn der Client am Authorization Endpunkt einen Pairwise Sub Claim anfragt
  ```

Scopes

- conf/attribute-filter.xml

```
<AttributeFilterPolicy id="OPENID_SCOPE_PSEUDONYMOUS">
    <PolicyRequirementRule xsi:type="oidc:OIDCScope" value="pseudonymous" />
    <AttributeRule attributeID="subject-pairwise">
        <PermitValueRule xsi:type="ANY" />
    </AttributeRule>
    <AttributeRule attributeID="eduPersonScopedAffiliation">
        <PermitValueRule xsi:type="ANY" />
    </AttributeRule>
    <!-- u.s.w. -->
</AttributeFilterPolicy>
```

Optionale Tokenverschlüsselung

- unsere Doku
- Fehlerquelle! → erst ohne Tokenverschlüsselung zum Laufen bringen
- Anschalten in `conf/oidc.properties`
- Anschalten im OIDC.SSO-Profil in `conf/relying-party.xml`
- bei manueller Client-Reg. Auffüllen der RP-Metadaten auf dem OP mit Infos zu Schlüsselmaterial und Signaturalgorithmen
- zurück auf Konfiguration ohne Token-Verschlüsselung: auf beiden Seiten die Verschlüsselungsalgorithmen aus Metadaten entfernen

Discovery und Client Registrierung

Manuelle Client-Registrierung

- Hinterlegen von RP-Metadaten auf OP
- mindestens Client ID, Client Secret, Redirect URI
- Option 1: gewohntes Einbinden von xml-Metadaten
  - RP-Metadaten im xml-Format
  - Einbinden in `conf/metadata-providers.xml`
- Option 2: JSON-Metadaten
  - RP-Metadaten im JSON-Format
  - entspr. FileResolver in `conf/oidc-clientinfo-resolvers.xml` aktivieren

Manuelle Client-Registrierung in der VM

- idp1.local
  - `metadata/oidc-client.json` → JSON-Metadaten beider lokalen Rps
  - `conf/relying-party.xml` → OIDC.Registration ist nicht konfiguriert
- sp1.local
  - `/etc/apache2/sites-enabled/sp1.conf`
  - URL der OP-Metadaten ist hart verdrahtet (*ein* OP)
  - Client ID und Client Secret sind fest vergeben

Dynamische Client-Registrierung

- conf/oidc.properties → dynreg-Einstellungen vornehmen
  - Speicherdauer der Registrierung
  - Speicherort der registrierten Client-Informationen (z.B. DB)
  - akzeptierte Scopes
  - Standardwert für Sub Claim
- conf/relying-party.xml: Profil `OIDC.Registration` für unverifizierte RP erlauben

Vielen Dank! Gibt‘s Fragen?

- Kontakt
  - DFN-AAI Team

    E-Mail: hotline@aai.dfn.de
    Tel.: +49-30-884299-9124
    Fax: +49-30-884299-370

    Anschrift:
    DFN-Verein, Geschäftsstelle
    Alexanderplatz 1
    10178 Berlin