Microsoft ADFS2 im DFN AAI

Microsofts ADFS ist leider nicht in der Lage mehrere Relying Parties (oder Service-Provider, wie sie im Shibboleth2-Sprachgebrauch heißen) in einer einzigen Metadaten-Datei zu importieren. Von daher bietet es sich an entweder FEMMA zu nutzen (welches seit einiger Zeit nicht mehr weiterentwickelt wird und nach wie vor ein Beta-Produkt ist) oder eine „Brücke“ zur Shibboleth-Welt zu bauen.

Der ADFS2-Server unterstützt netterweise das SAML-Protokoll, so dass es durchaus möglich ist den ADFS als IdP zu benutzen.
Als IdP für das DFN dient am besten ein Shibboleth IdP, da die Installation von selbigem auf den Seiten des DFN AAI gut dokumentiert ist. In Verbindung mit einem Shibboleth SP, wird eine Brücke zum ADFS daraus.

Der SP arbeitet durchaus auch mit IIS zusammen und auch Tomcat lässt sich auf dem IIS betreiben, allerdings ist selbiges nicht für den Einsatz als Brücke geeignet, da der IIS die Remote-Authentifizierung über den ISAPI-Filter der Apache-Foundation nicht an Tomcat weitergeben kann.
Die hier vorgestellte Brückentechnologie arbeitet ausschließlich unter Windows.

Installation

Folgende Komponenten werden benötigt:

- Apache Webserver (Apache)
- Apache Tomcat (Tomcat) + JDK
- Shibboleth Service Provider (SP)
- Shibboleth Identity Provider (IdP)

Konfiguration

Die Konfiguration richtet sich nach den Standardeinstellungen. Diese Auflistung stellt keinen Anspruch an Vollständigkeit, zeigt aber einige Stellen auf, die besonders hervorzuheben sind.

Apache Webserver

Standardmäßig werden in Apache alle Dateien des Formats *.conf aus dem „conf/extra“-Ordner geladen. Dort wurden folgende Dateien hinzugefügt:

conf/extra/Shibboleth-idp.conf

```
##############################################
#
# SingleSignOnService
#

Listen 443
```

```
<VirtualHost _default_:443>
  #   General setup for the virtual host
  [...]

  <Proxy ajp://localhost:8009/idp/*>
    Allow from all
  </Proxy>
  ProxyPass /idp/ ajp://localhost:8009/idp/
</VirtualHost>


##############################################
#
# ArtifactResolutionService und AttributeService
#

Listen 8443

<VirtualHost _default_:8443>
  #   General setup for the virtual host
  [...]
  <Proxy ajp://localhost:8009/idp/*>
    Allow from all
  </Proxy>
  ProxyPass /idp/ ajp://localhost:8009/idp/
</VirtualHost>
```

conf/extra/Shibboleth-sp.conf

```
#Include Shibboleth SP Configuration

Include "[IHR SP-Pfad]/etc/shibboleth/apache22.config

<LocationMatch "(/idp/(.*)/SSO)|(/idp/Authn)">
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</LocationMatch>
```

Apache Tomcat

Es können im Prinzip alle Connectoren auskommentiert werden es muss aber der Authentifizierungsmechanismus für Port 8009 gesetzt werden:

conf/Server.xml

```
[...]
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
tomcatAuthentication="false" />
[...]
```

Shibboleth-IdP

Es empfiehlt sich, dass der IdP die Daten die er übermitteln soll selbst lädt. Da sich ADFS auf eine AD bezieht wird hier ebenfalls die AD benutzt. Außerdem muss der IdP auf den Remote-Login-Handler umgestellt werden, so dass die Authentifizierung von Apache durchgeführt wird.

conf/attribute-resolver.xml

```
[…]
   <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
      ldapURL="ldap://dc-01 ldap://dc-02 ldap://dc-03"
      baseDN="DC=Uni-Mainz,DC=DE"


principal="[Benutzer mit LDAP-Zugriff]"


principalCredential="[Passwort]"


connectionStrategy="ROUND_ROBIN">

       <dc:FilterTemplate>
          <![CDATA[
             (userPrincipalName=$requestContext.principalName)
          ]]>
       </dc:FilterTemplate>

     <dc:LDAPProperty name="java.naming.referral" value="follow"/>
   </resolver:DataConnector>
[…]
```

conf/handler.xml

```
[…]
   <!-- Login Handlers -->
   <ph:LoginHandler xsi:type="ph:RemoteUser">
     <ph:AuthenticationMethod>
       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
     </ph:AuthenticationMethod>
       <ph:AuthenticationMethod>
       urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified
     </ph:AuthenticationMethod>
   </ph:LoginHandler>
[…]
```

Shibboleth-SP

Der Shibboleth-SP benötigt ebenfalls ein wenig zusätzliche Konfiguration, um ADFS Metadaten ohne weitere Änderungen zu unterstützen:

/etc/shibboleth/shibboleth2.xml

```
[…]
<ApplicationDefaults entityID=https://shib.uni-mainz.de/shibboleth
    REMOTE_USER="eppn persistent-id targeted-id"
    attributePrefix="AJP_">
[…]
  <MetadataProvider type="XML"
    uri=https://adfs/ FederationMetadata.xml
    backingFilePath=" FederationMetadata.xml" reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil"
      maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="fedsigner.pem"/>
  </MetadataProvider>
[…]
```

/etc/shibboleth/atribute-policy.xml

ADFS unterstützt von Haus aus keine Scopes, da aber das manuelle anpassen der ADFS-Metadaten eher mühselig ist und dazu führt, dass die Daten nicht mehr signiert sind, kann man den SP anweisen, den Scope nicht aus den Metadaten abzuleiten. Dazu ersetzt man die ScopingRules:

```
[…]
<afp:PermitValueRule id="ScopingRules" xsi:type="AND">
    <Rule xsi:type="NOT">
      <Rule xsi:type="AttributeValueRegex" regex="@"/>
    </Rule>
    <Rule xsi:type="AttributeScopeString" value="uni-mainz.de"/>
</afp:PermitValueRule>
[…]
```

ADFS Claim Rules

Zunächst extrahiert man die UPN aus der LDAP und sendet sie als UPN Claim (Send LDAP-Attribute as Claim).

Die folgende Regel ist noch hinzuzufügen, um die UPN als EPPN zu senden, damit der SP selbiges als Remote-User akzeptiert:

```
c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"]
 => issue(Type = "urn:oid:1.3.6.1.4.1.5923.1.1.1.6", Value = c.Value,
Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/attri
butename"] = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri");
```

Rückfragen

Bei Fragen zur Konfiguration bezüglich dieser ADFS-Shibboleth-Brücke können Sie gerne Kontakt aufnehmen mit glatzert@uni-mainz.de