Thüringer Universitäts- und Landesbibliothek
Jena

Shibboleth kommt zu MIR

Daniel Kirst

6. Januar 2015

Inhaltsverzeichnis

1 Das Grundsystem von Shibboleth **1**
- 1.1 Grundlagen der Kommunikation . . . 1
- 1.2 Der Identity Provider . . . 2
  - 1.2.1 Voraussetzungen . . . 2
  - 1.2.2 Installation . . . 2
  - 1.2.3 Einstellungen am Webserver . . . 3
  - 1.2.4 Konfigurieren von Tomcat . . . 4
  - 1.2.5 IdP-spezifische Einstellungen . . . 5
  - 1.2.6 Einlesen der SP Metadaten . . . 9
- 1.3 Der Service Provider . . . 9
  - 1.3.1 Voraussetzungen für diese Dokumentation . . . 9
  - 1.3.2 SSL für Apache einrichten . . . 9
  - 1.3.3 Installation . . . 10
  - 1.3.4 Einrichtung . . . 11
  - 1.3.5 Freigabe von Ressourcen . . . 13

2 Integration von Shibboleth in MIR **14**
- 2.1 Voraussetzung . . . 14
- 2.2 Das Login-Servlet . . . 14
- 2.3 Bereitstellung des Servlets . . . 14
- 2.4 Konfigurieren des Servlet-Containers . . . 15
- 2.5 Schützen des Login-Servlets . . . 16

3 Kurzübersicht **17**
- 3.1 Identity Provider . . . 17
  - 3.1.1 Wichtige Ordner/Dateien . . . 17
  - 3.1.2 Attribut-Freigabe des IdP . . . 17
  - 3.1.3 Neustarten . . . 18
  - 3.1.4 LDAP-Anfragen manuell testen . . . 18
- 3.2 Service Provider . . . 18
  - 3.2.1 Wichtige Dateien . . . 18
  - 3.2.2 Beispiel für Attribut-Freigabe des SP . . . 18
- 3.3 Apache Webserver . . . 18
  - 3.3.1 Wichtige Dateien/Einstellungen . . . 18
  - 3.3.2 Schutz von Ressourcen/Servlets . . . 19
- 3.4 Testen der Anwendung mit MIR . . . 20

Vorwort

Die nachfolgende Dokumentation fasst die wichtigsten Schritte für die Integration von *Shibboleth*[1] in die Repository-Anwendung *MIR*[2] zusammen. Bei Shibboleth handelt es sich um ein sogenanntes *Single Sign-On* (SSO)-System, welches es Nutzern nach einer einmaligen Authentifizierung ermöglicht auf geschützte Ressourcen unterschiedlicher Systeme zuzugreifen ohne sich bei jedem dieser Systeme einzeln anzumelden.

Für die Realisierung eines solchen Vorhabens liefert Shibboleth zwei Kernkomponenten aus: den *Service Provider* (SP) und den *Identity Provider* (IdP). Beide werden in dem *Abschnitt 1 Das Grundsystem von Shibboleth* vorgestellt. Dabei wird auf die jeweilige Installation und Konfiguration der Komponenten und ihrer Abhängigkeiten eingegangen.

Unter *Abschnitt 2 Integration von Shibboleth in MIR* dieser Dokumentation befindet sich eine schrittweise Anleitung, wie Shibboleth mit MIR verbunden werden kann, sodass Anwender sich zunächst über Shibboleth authentifizieren müssen, bevor sie in MIR auf Ressourcen zugreifen können.

1 Das Grundsystem von Shibboleth

1.1 Grundlagen der Kommunikation

Wie bereits erwähnt besteht ein Shibboleth System aus einem Identity- und Service Provider. Die Hauptaufgabe eines SP ist es die Ressourcen auf einem System vor unerlaubtem Zugriff zu schützen. Bei einer Ressourcen kann es sich um eine Webseite, einen Dienst oder eine Datei handeln, die nicht allen Nutzern zugänglich gemacht werden soll.

Greift ein Anwender auf eine geschützte Ressource zu, wird er zunächst per HTTP-Redirect an einen IDP weitergeleitet. Das Ziel des SP ist es dabei, den Benutzer sich an einen ihm bekannten IdP anmelden zu lassen, um dessen Identität zu verifizieren. Ein IdP ist somit ein System, welches auf Anfragen von SP wartet, um Nutzer zu authentifizieren (mehr dazu im *Abschnitt Der Identity Provider*).

Hat sich ein Anwender erfolgreich angemeldet wird er mittels HTTP-Redirect zurück zu dem SP geleitet. In dem Redirect werden Attributinformationen zu dem Benutzer übertragen, wie beispielsweise in welcher Rolle er beim IdP angemeldet ist (Mitarbeiter, Gast, Student ...). Diese Informationen wertet der Service Provider aus, um zu entscheiden, ob der Nutzer auf die angeforderte Ressource zugreifen darf.

Der Austausch der Informationen zwischen SP und IdP erfolgt über sogenannte *Assertions*, welche von der *Security Assertion Markup Language* (SAML)[3] bereitstellt werden. Sie bilden die Grundlage einer jeden Kommunikation und bestehen jeweils aus einem SAML-Request (vom SP) und einem SAML-Response (vom IdP). Die *Abbildung 1 Ablauf einer Anfrage* stellt den eben beschriebenen Vorgang grafisch dar.

[1] https://shibboleth.net/
[2] http://www.mycore.de/mir/content/index.xml
[3] XML-Framework zum Austausch sicherheitsrelevanter Nachrichten über Netzwerke

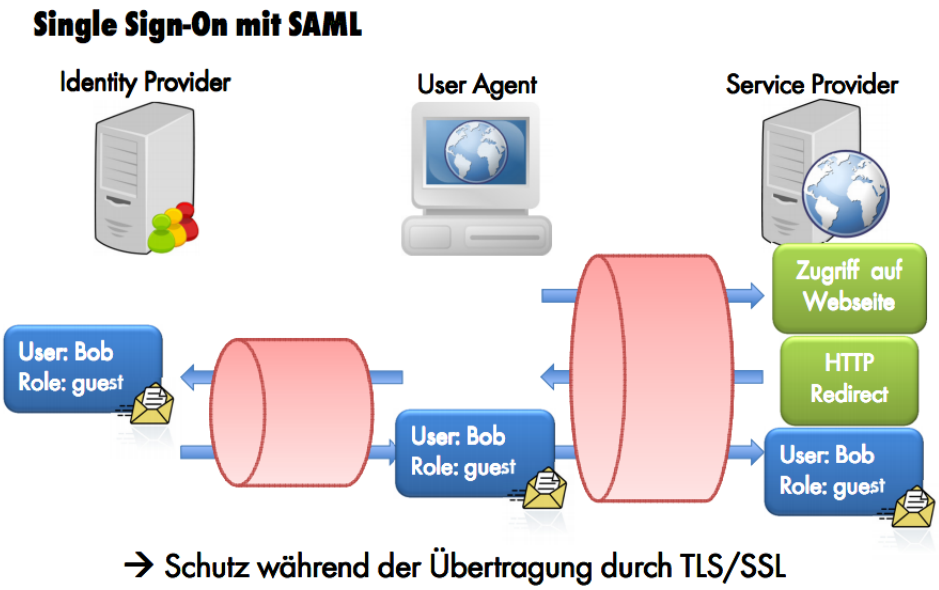

Abbildung 1: Ablauf einer Anfrage

1.2 Der Identity Provider

Die Aufgabe eines Identity Providers ist es, Anfragen von Service Providern entgegenzunehmen und Benutzer mit Hilfe eines bekannten Systems zu authentifizieren. Nach erfolgreicher Authentifizierung werden Attributinformationen zu dem Benutzer abgefragt, die zusammen mit ihm zurück zum Service Provider geleitet werden.

1.2.1 Voraussetzungen

In den folgenden Abschnitten wird erklärt, wie ein Shibboleth Identity Provider in der Version 2.4.0 auf einem Rechner mit Cent OS 6.5 installiert und eingerichtet werden kann. Ein lauffähiger Apache Webserver mit aktivem SSL-Modul, eine funktionierende Apache Tomcat Installation (Version 7.0.47) und Java werden vorausgesetzt.

Sowohl die Authentifizierung als auch die Attributabfrage erfolgt über den Verzeichnisdient des Rechenzentrums der Universität Jena, auf den mit LDAP zugegriffen wird.

1.2.2 Installation

Zunächst muss der Identity Provider heruntergeladen und entpackt werden. Sämtliche Versionen der Identity Provider werden auf der offiziellen Shibboleth-Webseite zum Download[4] zur Verfügung gestellt.

[4] http://shibboleth.net/downloads/identity-provider/2.4.0/shibboleth-identityprovider-2.4.0-bin.tar.gz

Anschließend wird der beigelegte Installer ausgeführt, der den Nutzer durch die Installation begleitet. Während der Installation muss der *fully qualified domain name (FQDN)* des IdP angegeben werden. Um die Abwärtskompatibilität mit älteren Service Providern zu gewährleisten, wird empfohlen *keine Großbuchstaben* in dem Namen zu verwenden. In dieser Installation wird der Identity Provider unter `/opt/shibboleth-idp3` installiert, als FQDN wird `https://shib-test.thulb.uni-jena.de` verwendet.

```
[root@shib-test ~]# wget http://shibboleth.net/downloads/identity-\
                        provider/latest/shibboleth-identityprovider-\
                        2.4.0-bin.zip
[root@shib-test ~]# unzip shibboleth-identityprovider-2.4.0-bin.zip
[root@shib-test ~]# cd shibboleth-identityprovider-2.4.0
[root@shib-test ~]# ./install.sh
```

1.2.3 Einstellungen am Webserver

Zunächst muss der Webserver konfiguriert werden. Die zentrale Konfigurationsdatei von *httpd* heißt *httpd.conf* und liegt im Verzeichnis `/etc/httpd/conf`. Es hat sich als gute Praxis erwiesen, diese Datei nicht vollständig zu modifzieren, sondern eine externe Konfigurationsdatei zu erstellen. Diese wird anschließend in *httpd.conf* eingebunden und somit geladen.

Damit der Austausch der Daten zwsichen einem IdP und einem SP verschlüsselt stattfindet, wird das SSL-Modul von *httpd* verwendet, dessen Konfigurationsdatei sich in `/etc/httpd/conf.d/` befindet und *ssl.conf* heißt. Durch die Installation des IdP ist ein Zertifikat (idp.crt) und ein privater Schlüssel (idp.key) in dem Verzeichnis `/opt/shibboleth-idp3/credentials/` generiert worden. Diese müssen in der Konfigurationsdatei angegeben werden. Der Webserver muss nicht zwangsweise ein Certificate Chain File besitzen - diese Zeile kann mit einer Raute auskommentiert werden. Es folgt ein Ausschnitt der Konfigurationsdatei und den wichtigsten Einstellungen.

```
Listen 443
<VirtualHost 141.35.20.206:443>
...
ServerName               shib-test.thulb.uni-jena.de:443
SSLEngine                on
SSLCertificateFile       /opt/shibboleth-idp3/credentials/idp.crt
SSLCertificateKeyFile    /opt/shibboleth-idp3/credentials/idp.key
#SSLCertificateChainFile /opt/shibboleth-idp3/credentials/xyz.ca
...
    <Location /idp>
      Allow from all
      ProxyPass ajp://localhost:8009/idp
    </Location>
</VirtualHost>
...
```

```
Listen 8843
<VirtualHost 141.35.20.206:8443>

ServerName                shib-test.thulb.uni-jena.de:8443
SSLEngine                 on
SSLCertificateFile        /opt/shibboleth-idp3/credentials/idp.crt
SSLCertificateKeyFile     /opt/shibboleth-idp3/credentials/idp.key
SSLVerifyClient           optional_no_ca
SSLVerifyDepth            10
SSLOptions                +StdEnvVars +ExportCertData
    <Location /idp>
      Allow from all
      ProxyPass ajp://localhost:8009/idp
    </Location>
</VirtualHost>
```

Nachdem die Einstellungen geändert worden sind, sollte der Webserver neugestartet werden. Die Funktionstüchtigkeit der eben übernommen Änderungen kann anschließend durch einen Aufruf von https://localhost getestet werden.

```
[root@shib-test ~]# /etc/init.d/httpd restart
```

1.2.4 Konfigurieren von Tomcat

An dieser Stelle ist der Webserver einsatzbereit und es folgt die Konfiguration von *Tomcat*. Dazu muss die Datei *server.xml* editiert werden, welche sich bei dieser Installation in dem Verzeichnis /usr/local/apache-tomcat-7.0.47/conf befindet. In ihr muss der AJP-Connector konfiguriert werden, damit der Webserver mit Tomcat kommunizieren kann. Folgender Ausschnitt sollte sich in der Datei befinden:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009"
        address="127.0.0.1"
         protocol="AJP/1.3"
         redirectPort="8443"
         enableLookups="false"
         tomcatAuthentication="false" />
```

Anschließend muss noch die IdP-Webanwendung in Tomcat aktiviert werden. Dazu wird in dem Verzeichnis /usr/local/apache-tomcat-7.0.47/conf/Catalina/localhost eine Datei namens *idp.xml* mit folgendem Inhalt erzeugt:

```
<Context docBase="/opt/shibboleth-idp3/war/idp.war"
         privileged="true"
         antiResourceLocking="false"
         antiJARLocking="false"
         unpackWAR="false"
```

```
        swallowOutput="true"
        cookies="false" />
```

Nach diesen Änderungen sollte Tomcat neugestartet werden. Hat Tomcat die Initialisierung des neuen Servlets durchgeführt, kann mit einem Aufruf der Seite https://shib-test.thulb.uni-jena.de/idp/profile/Status getestet werden, ob der IdP erfolgreich läuft.

```
[root@shib-test ~]# /usr/local/apache-tomcat-7.0.47/bin/shutdown.sh
[root@shib-test ~]# /usr/local/apache-tomcat-7.0.47/bin/startup.sh
```

1.2.5 IdP-spezifische Einstellungen

Sämtliche Konfigurationsdateien, die direkt den IdP betreffen, befinden sich in dem Installationsverzeichnis /opt/shibboleth-idp3/conf/. Da die Konfigurationsoptionen teilweise recht komplex sind, bezieht sich dieser Abschnitt nur auf die wesentlichsten Änderungen, die für einen Betrieb des IdP notwendig sind.

Nachdem Änderungen an dem IdP durchgeführt worden sind, sollte Tomcat neugestartet werden. Es empfiehlt sich daher, erst alle Änderungen zu tätigen und anschließend einen Neustart durchzuführen.

logging.xml In dieser Datei werden die Einstellungen für das Logging des Systems verwaltet. Es wird empfohlen, das Logging-Level von *LDAP*, *OpenSAML* und dem *IdP* auf *DEBUG* zu stellen, um schneller Fehler in den Logs ausfindig zu machen.

Die Log-Dateien befinden sich in dem Verzeichnis /opt/shibboleth-idp3/logs. Die wichtigste Datei ist dabei *idp-process.log*, die bei Fehlerfällen als erstes konsultiert werden sollte.

```
<!-- Logs IdP, but not OpenSAML, messages -->
<logger name="edu.internet2.middleware.shibboleth" level="DEBUG"/>

<!-- Logs OpenSAML, but not IdP, messages -->
<logger name="org.opensaml" level="DEBUG"/>

<!-- Logs LDAP related messages -->
<logger name="edu.vt.middleware.ldap" level="DEBUG"/>
```

login.config Diese Datei beinhaltet die Art und Weise der Authentifizierung von Benutzern. Dabei gibt es mehrere Möglichkeiten, wie sich Anwender authentifizieren können. Mit Hilfe eines gewöhnlichen Logins, bestehend aus einer BenuzerID und einem Passwort, kann sich der Anwender gegen eine LDAP-Datenbank, gegen Active Directory oder gegen einen Kerberos-Server authentifizieren.

Darüber hinaus gibt es noch weitere Varianten, die allerdings hier nicht näher erläutert werden. In dieser Beispielinstallation sollen sich Nutzer mittels LDAP authentifizieren. Eine Beispielkonfiguration sieht wie folgt aus:

```
ShibUserPassAuth {

   edu.vt.middleware.ldap.jaas.LdapLoginModule required
      ldapUrl="pathToLDAP"
      tls="true"
      bindDn="uid=hiddenUIDForLDAP,ou=local,dc=uni-jena,dc=de"
      bindCredential="***hiddenPassword***"
      baseDn="ou=users,dc=uni-jena,dc=de"
      userFilter="uid={0}";
};
```

handler.xml Diese Datei spezifiziert die Login-Methode, die der IdP verwenden soll. Da das Login mittels Benutzername und Passwort selbst durchgeführt werden soll, muss der Part vom Typ *RemoteUser* auskommentiert werden.

```
<!-- Login Handlers -->
<!--
<ph:LoginHandler xsi:type="ph:RemoteUser">
    <ph:AuthenticationMethod>
        urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified
    </ph:AuthenticationMethod>
</ph:LoginHandler>
-->
<!--  Username/password login handler -->
<ph:LoginHandler xsi:type="ph:UsernamePassword"
 jaasConfigurationLocation="file:///opt/shibboleth-idp3/conf/
 login.config">
    <ph:AuthenticationMethod>
        urn:oasis:names:tc:SAML:2.0:ac:classes:
        PasswordProtectedTransport
    </ph:AuthenticationMethod>
</ph:LoginHandler>
```

relying-party.xml Der Inhalt dieser Datei bestimmt das Verhalten beim Umgang mit Metadaten. In ihr müssen die *MetadataProvider* eingetragen werden und sogenannte *TrustEngines* definiert werden.

Bei einer *TrustEngine* handelt es sich um die zentrale Verwaltungseinheit des IdP Sicherheitssystems. Über sie wird geregelt, wie der gesicherte Austausch von Metadaten und Assertions erfolgt.

Dazu benutzt die TrustEngine das bei der Installation angelegte Zertifikate *idp.crt* und den private Schlüssel *idp.key* wieder. Folgende Zeilen sollten in der Datei enthalten sein:

```
<rp:AnonymousRelyingParty provider=
   "https://shib-test.thulb.uni-jena.de/idp/shibboleth"
    defaultSigningCredentialRef="IdPCredential"/>
<metadata:MetadataProvider id="ShibbolethMetadata"
 xsi:type="metadata:ChainingMetadataProvider">
```

```
...
<!-- Load the IdP's own metadata for artifact support. -->
<metadata:MetadataProvider id="IdPMD"
 xsi:type="metadata:FilesystemMetadataProvider"
 metadataFile="/opt/shibboleth-idp3/metadata/idp-metadata.xml"
 maxRefreshDelay="P1D" />

    <!-- Important SP Metadata -->
    <metadata:MetadataProvider
     id="IdSP" xsi:type="metadata:FilesystemMetadataProvider"
     metadataFile="/opt/shibboleth-idp3/metadata/sp-metadata.xml"
     maintainExpiredMetadata="true"/>
...
<security:Credential id="IdPCredential" xsi:type="security:X509Filesystem">
    <security:PrivateKey>
         /opt/shibboleth-idp3/credentials/idp.key
    </security:PrivateKey>
    <security:Certificate>
         /opt/shibboleth-idp3/credentials/idp.crt
    </security:Certificate>
</security:Credential>

<!-- Trust engine used to evaluate the signature on loaded metadata. -->
<security:TrustEngine id="shibboleth.MetadataTrustEngine"
 xsi:type="security:StaticExplicitKeySignature">
        <security:Credential id="MyFederation1Credentials"
         xsi:type="security:X509Filesystem">
            <security:Certificate>
                  /opt/shibboleth-idp3/credentials/idp.crt
            </security:Certificate>
        </security:Credential>
</security:TrustEngine>
```

attribute-resolver.xml Diese Datei gibt an, welche Attribute zu einem Nutzer abgefragt werden sollen. Dabei muss jedes Attribut zuvor mit Hilfe eines Datentyps definiert werden. Zur Abfrage der Daten wird ein LDAP Connector mit der Bezeichnung *myLDAP* erzeugt.

Das folgende Beispiel zeigt an, wie die User-ID und die E-Mail-Adresse eines Nutzers abgefragt werden können:

```
<resolver:AttributeDefinition xsi:type="ad:Simple" id="uid"
 sourceAttributeID="uid">
  <resolver:Dependency ref="myLDAP" />
   <resolver:AttributeEncoder xsi:type="enc:SAML1String"
    name="urn:mace:dir:attribute-def:uid" />
   <resolver:AttributeEncoder xsi:type="enc:SAML2String"
    name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid" />
</resolver:AttributeDefinition>
```

```
<resolver:AttributeDefinition xsi:type="ad:Simple" id="email"
 sourceAttributeID="mail">
  <resolver:Dependency ref="myLDAP" />
  <resolver:AttributeEncoder xsi:type="enc:SAML1String"
   name="urn:mace:dir:attribute-def:mail" />
  <resolver:AttributeEncoder xsi:type="enc:SAML2String"
   name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail" />
</resolver:AttributeDefinition>

<resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="urlToLDAP"
        baseDN="ou=users,dc=uni-jena,dc=de"
        useStartTLS="true"
        principal="uid=ldapreader,ou=local,dc=uni-jena,dc=de"
        principalCredential="***hiddenPassword***">
        <dc:FilterTemplate>
            <![CDATA[
                (uid=$requestContext.principalName)
            ]]>
        </dc:FilterTemplate>
</resolver:DataConnector>
```

attribute-filter.xml Diese Datei fungiert als eine Art Gebotsfilter, der angibt, welche Attribute aus der Datei *attribute-resolver.xml* an welchen SP übergeben werden dürfen. Das folgende Beispiel zeigt die Freigabe der User-ID für jeden Service Provider:

```
<afp:AttributeRule attributeID="uid">
       <afp:PermitValueRule xsi:type="basic:ANY" />
</afp:AttributeRule>
```

service.xml Durch das Hinzufügen des XML-Attributs *configurationResourcePollingFrequency* bei den Attribut-Resolvern, wird der IdP angewiesen, in regelmäßigen Intervallen die Dateien *attribute-filter.xml* und *attribute-resolver.xml* automatisch nachzuladen. Der Wert *PT15M* gibt dabei an, dass die Aktualisierung aller 15 Minuten stattfindet.

```
<srv:Service id="shibboleth.AttributeResolver"
 xsi:type="attribute-resolver:ShibbolethAttributeResolver"
 configurationResourcePollingFrequency="PT15M">
 <srv:ConfigurationResource file="/opt/shibboleth-idp3/conf/
 attribute-resolver.xml" xsi:type="resource:FilesystemResource"/>
</srv:Service>

<srv:Service id="shibboleth.AttributeFilterEngine"
 xsi:type="attribute-afp:ShibbolethAttributeFilteringEngine"
 configurationResourcePollingFrequency="PT15M">
    <srv:ConfigurationResource file="/opt/shibboleth-idp3/conf/
     attribute-filter.xml" xsi:type="resource:FilesystemResource"/>
</srv:Service>
```

1.2.6 Einlesen der SP Metadaten

An dieser Stelle sind alle Konfigurationsdateien editiert worden und der IdP ist fast funktionsbereit. Lediglich die Metadaten des Service Providers fehlen.

Der Leser ist dazu angehalten, zunächst einen Service Provider anzulegen, bevor er an dieser Stelle dessen Metadaten beziehen kann. Es ist darauf zu achten, dass die neuesten Metadaten heruntergeladen werden und an den selben Ort gespeichert werden, wie er in der Datei *relying-party.xml* angegeben ist.

Bei dieser Beispielinstallation befinden sich die SP Metadaten mit der Bezeichnung *sp-metadata.xml* in dem Ordner `/opt/shibboleth-idp3/metadata/`. Sind alle Einstellungen vorgenommen worden sollte Tomcat erneut gestartet werden. Der Identity Provider ist nun einsatzbereit.

```
[root@shib-test ~]# curl -k https://shibsp-test.thulb.uni-jena.de/
                    Shibboleth.sso/Metadata > shibsp-test-metadata-backingFile.xml
[root@shib-test ~]# /usr/local/apache-tomcat-7.0.47/bin/shutdown.sh
[root@shib-test ~]# /usr/local/apache-tomcat-7.0.47/bin/startup.sh
```

1.3 Der Service Provider

Ein Shibboleth Service Provider besteht aus zwei Software-Komponenten: einem Daemon namens *shibd* und einem Software-Modul für den Apache Webserver. Der Daemon hat die Aufgabe Assertions zu erstellen und auszuwerten, wohingegen das Apache-Shibboleth-Modul als Schnittstelle zwischen Nutzeranfragen und shibd dient. Zusammen sorgen beide Elemente dafür Ressourcen auf dem Zielsystem zu schützen.

1.3.1 Voraussetzungen für diese Dokumentation

Der nachfolgende Text erklärt schrittweise das Vorgehen für die Installation und Einrichtung eines Sibboleth Service Providers unter Mac OS X 10.9.5 Mavericks. Diese Anleitung geht davon aus, dass auf dem Rechner ein Apache Webserver in der Version 2.2.26 unter dem Verzeichnis `/etc/apache2/` installiert ist.

1.3.2 SSL für Apache einrichten

Damit der Datenaustausch zwischen dem Benutzer und SP verschlüsselt stattfindet, muss das SSL-Modul in Apache von der Datei `/etc/apache2/httpd.conf` geladen werden. Zusätzlich muss die entsprechende Konfigurationsdatei eingebunden werden:

```
LoadModule ssl_module libexec/apache2/mod_ssl.so
Include /etc/apache2/extra/httpd-ssl.conf
```

An dieser Stelle ist es erforderlich einen privaten Schlüssel und ein Zertifikat für die SSL-Verbindung zu generieren. In dieser Dokumentation wird ein selbstsigniertes Zertifikat erstellt, obwohl es auch möglich ist, ein Zertifikat einer CA zu verwenden:

```
#Erzeugen eines privaten Schlüssels
user@mac:~$ mkdir /private/etc/apache2/ssl
user@mac:~$ cd /private/etc/apache2/ssl
user@mac:~$ openssl genrsa -des3 -out server.key 4096

#Erstellen eines Zertifikatantrags
user@mac:~$ openssl req -new -key server.key -out server.csr

#Selbstsigniertes Zertifikat erstellen
user@mac:~$ openssl x509 *req -days 365 -in server.csr \
                -signkey server.key -out server.crt

#Entfernen des Passworts
user@mac:~$ cp server.key server.key.orig
user@mac:~$ openssl rya -in server.key.orig -out server.key
%\end{lstlisting}
```

Sobald der Schlüssel und das Zertifikat erstellt worden sind müssen diese in der Datei `/etc/apache2/extra/httpd-ssl.conf` eingetragen werden. Zusätzlich muss die SSLEngine aktiviert werden:

```
SSLEngine on
SSLCertificateFile "/etc/apache2/ssl/server.crt"
SSLCertificateKeyFile "/etc/apache2/ssl/server.key"
```

1.3.3 Installation

1. Entwickler-Tools (XCode) installieren
   (https://developer.apple.com/xcode/downloads/)

2. Den Paketmanager Mac Port installieren
   (https://www.macports.org/install.php)

3. Den Shibboleth Service Provider über das Terminal installieren:

   ```
   port install curl +ssl
   port install shibboleth
   ```

   Nach der Installation befindet sich der Shibboleth Service Provider unter dem Verzeichnis `/opt/local/etc/shibboleth`.

4. Symbolische Links setzen

   ```
   ln -s /opt/local/etc/shibboleth /etc/shibboleth}
   ln -s /opt/local/var/log/shibboleth /var/log/shibboleth}
   ln -s /opt/local/etc/shibboleth/apache22.config \
         /etc/apache2/other/shibboleth.conf
   ```

Bei dem letzten Link wird eine Konfigurationsdatei vom Shibboleth-Verzeichnis in das Apache-Verzeichnis kopiert. In dieser Datei erfolgt später die Ressourcenfreigabe. Zusätzlich lädt die Datei das benötigte Apache-Shibboleth-Modul.

Es ist darauf zu achten, dass die für die Apache-Version entsprechende Konfigurationsdatei verlinkt wird. In diesem Beispiel wird deshalb die Datei apache22.config verwendet.

5. Hinzufügen des Daemons shibd zum Autostart

   ```
   launchctl load -Fw /Library/LaunchDaemons/org.macports.shibd.plist
   ```

An dieser Stelle ist der Service Provider installiert und kann über die beiden nachfolgenden Befehle gestartet oder gestoppt werden:

```
launchctl   load -F /Library/LaunchDaemons/org.macports.shibd.plist
launchctl unload -F /Library/LaunchDaemons/org.macports.shibd.plist
```

1.3.4 Einrichtung

Nach der Installation muss der Service Provider zunächst konfiguriert werden, bevor er Ressourcen beschützen kann. Von den meisten Konfigurationsdateien existiert ein Backup, welches sich an der Dateiendung **.dist* erkennen lässt. Mit ihm kann der Nutzer im Notfall den Ursprungszustand wiederherstellen, falls der SP aufgrund einer fehlerhaften Konfiguration den Dienst verweigert.

Die Hauptkonfiguration findet in der Datei `/etc/shibboleth/shibboleth2.xml` statt. In ihr wird unter anderem der Speicherort für die Metadaten sowie Verbindungsdetails zum IdP hinterlegt. Viele Einstellungen sind in der Datei bereits vorkonfiguriert, sodass in diesem Abschnitt nur auf notwendige Änderungen eingegangen wird, um das System zum laufen zu bringen.

Zunächst muss eine *entityID* für den SP festgelegt werden. Diese wird in den SAML-Request eingebunden und gibt somit die Adresse des Absenders (des Service Providers) an. Damit die Übertragung verschlüsselt stattfindet ist darauf zu achten als Protokoll https anzugeben.

```
<ApplicationDefaults entityID="https://ulbp2735.ulb.uni-jena.de"
 attributePrefix="AJP_" REMOTE_USER="eppn persistent-id targeted-id">
```

Anschließend erfolgen einige Einstellung zu sogenannten Sessions. Eine Session wird vom SP dann angelegt, wenn er einen SAML-Response vom IdP mit Benutzerdaten erhält. Es kann festgelegt werden, wie lange eine Session pro Nutzer am System gültig ist und ob die Übertragung verschlüsselt stattfindet soll.

```
<Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
 checkAddress="false" handlerSSL="true" cookieProps="https">
```

In dem folgenden Tag wird die Adresse des Empfängers (IdP) angegeben. Zusätzlich wird festgelegt, welche SAML-Version zur Kommunikation verwendet werden kann. Dies

ist insbesondere hinsichtlich der Abwärtskompatiblität wichtig, falls ein SP mit einem älteren IdP kommunizieren möchte.

```
<SSO entityID="https://shib-test.thulb.uni-jena.de/idp/shibboleth">
     SAML2 SAML1
</SSO>
```

Um den Status eines Service Providers abzufragen müssen die berechtigten IP-Adressen über eine ACL eingetragen werden.

```
<Handler type="Status" Location="/Status" acl="127.0.0.1 ::1
 141.35.???.???"/>
```

Ein weiterer wichtiger Punkt in der Konfiguration der Datei shibboleth2.xml ist die Festlegung des MetadataProviders. Anhand der Metadaten weiß sowohl der IdP als auch der SP was für Daten er zu erwarten hast und wie die Kommunikation stattfindet.

In dem Beispiel wird ein MetadataProvider angelegt, der die Metadaten in einer XML-Datei abspeichert. Bei der Festlegung eines FileMetaDataProviders ist darauf zu achten, dass shibd lesende und schreibende Zugriffsrechte auf das angegebene Verzeichniss hat.

```
<MetadataProvider file="/Users/daniel/Shib-Cache/idp-metadata.xml"
 type="XML" />
```

Damit der Inhalt von Assertions ver- und entschlüsselt wird muss über einen *Credential-Resolver* festgelegt werden, wo sich der Schlüssel und das dazugehörige Zertifkat befinden. Für den SP aus diesem Beispiel wird das zuvor erzeugte Zertifikat für Apache und der entsprechende Schlüssel verwendet.

```
<CredentialResolver type="File" key="/etc/apache2/ssl/server.key"
 certificate="/etc/apache2/ssl/server.crt" />
```

Als nächstes muss die Datei `/etc/shibboleth/attribute-map.xml` editiert werden. Sie beinhaltet ein Mapping, welches festlegt, welche Attribute vom IdP empfangen werden und wie diese abgebildet sind.

Der Nutzer kann an der Stelle einfach alle Werte auskommentieren, die er nicht benötigt. Soll der SP etwa die E-Mail-Adresse als Attribut aus einem SAML-Response auslesen, sieht die Definition in dem Mapping wie folgt aus:

```
<Attribute name="urn:mace:dir:attribute-def:mail"   id="mail"/>
<Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

Nachdem alle Änderungen übernommen wurden muss der Webserver und shibd neugestartet werden (siehe Installation Schritt 5). Anschließend kann mit dem folgenden Befehl die Konfiguration auf Fehler geprüft und die Metadaten des IdP beschafft werden:

```
shibd -t
```

Ist die Konfiguration soweitgehend korrekt, dass der SP mit dem IdP kommunizieren kann sollte als Ergebnis im Terminal der Text *overall configuration is loadable, check*

console for non-fatal problems angezeigt werden. Gleichzeitig werden die Metadaten des IdP heruntergeladen und in das zuvor angegebene Verzeichnis abgespeichert.

Tritt dennoch ein Fehler auf, sollte zunächst die Logdatei `/var/log/shibboleth/shibd.log` überprüft werden.

1.3.5 Freigabe von Ressourcen

Ist der Shibboleth Service Provider installiert und konfiguriert kann die Freigabe von Ressourcen vorgenommen werden. Die Freigabe erfolgt dabei in der Konfigurationsdatei `/etc/apache2/other/apacheXX.conf`, wobei XX für die jeweilige Version des Webservers steht.

Nach jeder Änderung an der Freigabe muss der Apache-Webserver neugestartet werden. Das nachfolgende Beispiel zeigt, wie eine einfache Textdatei (`http://localhost/123.txt`) als Ressource geschützt werden kann:

```
#laden des Shibboleth-Moduls für Apache
LoadModule mod_shib /opt/local/lib/shibboleth/mod_shib_22.so

<Location /Shibboleth.sso>
  Satisfy Any
  Allow from all
</Location>

<IfModule mod_alias.c>
  <Location /shibboleth-sp>
    Satisfy Any
    Allow from all
  </Location>
  Alias /shibboleth-sp/main.css /opt/local/share/shibboleth/main.css
</IfModule>

<Location /123.txt>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession 1
  require shib-session
</Location>
```

2 Integration von Shibboleth in MIR

Sobald die grundlegende Kommunikation zwischen Identity- und Service Provider funktioniert, können verschiedene Ressourcen aus MIR mit dem SP geschützt werden.

In diesem Abschnitt wird gezeigt, welche Schritte notwendig sind, um den SP aus dem vorhergehenden Kapitel so zu erweitern, dass er Ressourcen von MIR schützt.

Als Ressource wird ein Servlet angelegt, das der Nutzer erst öffnen kann, nachdem er sich über einen IdP authentifiziert hat.

2.1 Voraussetzung

Die Voraussetzung dieses Kapitels ist ein fertig eingerichtetes, lauffähiges MIR. Der nachfolgende Abschnitt behandelt nicht die Installation und Konfiguration von MyCoRe und MIR. Entsprechende Informationen können auf der Projektwebseite[5] nachgelesen werden.

Als Servlet-Container wird Apache-Tomcat in der Version 8 eingesetzt. Für die Kommunikation zwischen Webserver und Servlet-Container wird das Apache-Modul *mod_ajp* eingesetzt. Bei gestartetem Tomcat ist die MIR-Webseite unter der URL `https://ulbp2735.ulb.uni-jena.de/dev-mir/content/index.xml` erreichbar.

2.2 Das Login-Servlet

Das Login-Servlet (`MCRShibbolethLoginServlet`) ist Bestandteil des mycore-user2-Moduls und fungiert als Schnittstelle zwischen dem IdP und SP.

Möchte ein Benutzer das Servlet aufrufen, ist allerdings noch nicht am System angemeldet, wird ein SAML-Request an den IdP gestellt. Erst nach einer erfolgreichen Authentifikation wird der Code des Servlets ausgeführt.

Die Aufgabe des Servlets ist es die Attributinformationen aus dem HTTP-Response entgegenzunehmen und daraus eine MyCoRe-Session für den Benutzer zu erstellen. Bei diesem Vorgang werden vier wichtige Attribute ausgewertet: die *User-ID* (uid), der *Display-Name*, die *E-Mail* und *eduPersonAffiliation*. Der letzte Parameter beinhaltet die Rollenzugehörigkeit und legt fest, ob der Nutzer nur lesende oder auch schreibende Rechte in MIR erhält.

2.3 Bereitstellung des Servlets

Damit der Nutzer auf das Servlet zugreifen kann, muss es in die Datei `realms.xml` eingetragen werden. In ihr werden die bekannten Realms eingetragen, über die sich ein

[5] `http://www.mycore.de/documentation/getting_started/mir.html`

Nutzer anmelden kann.

```
#Inhalt der Datei reamls.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<realms local="local">
  <realm id="servlet3">
    <label xml:lang="de">IdP Login</label>
    <label xml:lang="en">IdP Login</label>
    <login url="MCRShibbolethLoginServlet" redirectParameter="url">
      <label xml:lang="de">Login-IdP</label>
      <label xml:lang="en">Login-IdP</label>
      <info>
        <label xml:lang="de">
          text
        </label>
        <label xml:lang="en">
          text
        </label>
      </info>
    </login>
  </realm>
</realms>
```

Zusätzlich muss das Servlet im Deployment-Descriptor (`web-fragment.xml`) hinzugefügt werden, um unter der angegebenen Adresse erreichbar zu sein.

Werden Änderungen am Servlet vorgenommen muss das entsprechende MyCoRe-Modul und die MIR-Webanwendung neu gebaut werden.

```
#Auszug aus der Datei web-fragment.xml
<servlet id="MCRShibbolethLoginServlet">
  <servlet-name>MCRShibbolethLoginServlet</servlet-name>
  <servlet-class>
    org.mycore.user2.login.MCRShibbolethLoginServlet
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>MCRShibbolethLoginServlet</servlet-name>
  <url-pattern>/servlets/MCRShibbolethLoginServlet</url-pattern>
</servlet-mapping>
```

2.4 Konfigurieren des Servlet-Containers

Dieses Beispiel verwendet Tomcat 8 als Servlet-Container. Die Kommunikation zwischen Servlet-Container und dem Webserver geschieht mit Hilfe des *Apache JServ Protocol* (AJP). Aus diesem Grund muss die Datei `conf/server.xml` im Tomcat-Installationsverzeichnis angepasst werden.

```
#Ausschnitt aus der Tomcat-Konfiguration server.xml
<Connector port="8009" address="ulbp2735.ulb.uni-jena.de"
 protocol="AJP/1.3" redirectPort="8443" enableLookups="false"
 tomcatAuthentication="false" />
```

2.5 Schützen des Login-Servlets

Damit das Servlet geschützt wird, muss die Datei /etc/apache2/other/apacheXX.conf um folgenden Eintrag ergänzt werden:

```
ProxyPass /dev-mir ajp://ulbp2735.ulb.uni-jena.de:8009/dev-mir

#schützt das Servlet vor unerlaubten Zugriffen
<Location /dev-mir/servlets/MCRShibbolethLoginServlet>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

3 Kurzübersicht

3.1 Identity Provider

3.1.1 Wichtige Ordner/Dateien

conf/attribute-resolver.xml	legt fest welche Attribute von LDAP geholt werden sollen, inkl. Datentypdefinition
conf/attribute-Filter.xml	welche Attribute sollen an den SP vermittelt werden
conf/login.conf	Einrichtung des ldap-Connectors
conf/relying-party.xml	Konfiguration vom Umgang mit Assertions, der Metadaten und von TrustEngines (Verschlüsselung)
conf/handler.xml	legt fest, welche Authentifizierungsmethoden der IDP unterstützt
/credentials	dient als Ablage für private Schlüssel, Zertifikate und Keystores, die dann in relying-party.xml referenziert werden
/logs/idp-process.log	diese Datei sollte als erstes bei auftretenden Fehlern oder Nichtfunktionalität geprüft werden
/metadata	Default-Speicherort für Metadaten, die in der Datei relying-party.xml aufgeführt sind

3.1.2 Attribut-Freigabe des IdP

Nach jeder Änderung in der Freigabe muss der Servlet-Container neugestartet werden.

```
#in der Datei attribute-resolver.xml
<resolver:AttributeDefinition xsi:type="ad:Simple" id="uid"
 sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
         name="urn:mace:dir:attribute-def:uid" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
         name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid" />
</resolver:AttributeDefinition>

#in der Datei attribute-filter.xml
<afp:AttributeRule attributeID="uid">
        <afp:PermitValueRule xsi:type="basic:ANY" />
</afp:AttributeRule>
```

3.1.3 Neustarten

```
#Herunterfahren
/usr/local/apache-tomcat-7.0.47/bin/shutdown.sh

#sichergehen, dass Servlet-Container beendet ist
ps -ef | grep tomcat

#Starten
/usr/local/apache-tomcat-7.0.47/bin/startup.sh
```

3.1.4 LDAP-Anfragen manuell testen

```
ldapsearch -x  -H ldaps://adresse.rz.uni-jena.de -D
uid=ldapreader,ou=local,dc=uni-jena,dc=de -w
passwort -b ou=users,dc=uni-jena,dc=de
```

3.2 Service Provider

3.2.1 Wichtige Dateien

attribute-map.xml	definiert ein Mapping für übermittelte Attribute des IdP
apacheXX.config	Konfigurationsdatei für den jeweiligen Apache-Webserver, die idie Freigabe von Ressourcen regelt
shibboleth2.xml	Die Hauptkonfiguration des SP wird in dieser Datei vorgenommen
shibd.log	Wichtigste Log-Datei für den Service-Provider

3.2.2 Beispiel für Attribut-Freigabe des SP

```
#in der Datei attribute-map.xml
<Attribute name="urn:mace:dir:attribute-def:displayName"
 id="displayName"/>
<Attribute name="urn:oid:2.16.840.1.113730.3.1.241"
 id="displayName"/>
```

Für die Freigabe von Attributen ist KEIN erneuter Austausch von Metadaten notwendig, allerdings das Neustarten von Tomcat, Apache und den SP! Nach Änderung der Freigabeoptionen genügt ein Neustarten des Apache Webservers.

3.3 Apache Webserver

3.3.1 Wichtige Dateien/Einstellungen

```
httpd.conf
* in dieser Datei muss die apacheXX.config des SP eingebunden
```

```
  werden (Include /private/etc/apache2/other/*.conf)
* mod_ssl, mod_proxy und mod_proxy_ajp müssen aktiviert sein
 (LoadModule proxy_ajp_module libexec/apache2/mod_proxy_ajp.so ...)

httpd-ssl.conf
* die SSLEngine muss den Wert on besitzen
* SSLCertificateFile muss auf das SP.crt zeigen
* SSLCertificateKeyFile muss auf den SP.key zeigen
```

3.3.2 Schutz von Ressourcen/Servlets

```
#laden des Apache-Moduls
LoadModule mod_shib /opt/local/lib/shibboleth/mod_shib_22.so

<Location /Shibboleth.sso>
  Satisfy Any
  Allow from all
</Location>

<IfModule mod_alias.c>
  <Location /shibboleth-sp>
    Satisfy Any
    Allow from all
  </Location>
  Alias /shibboleth-sp/main.css /opt/local/share/shibboleth/main.css
</IfModule>

#schützt den Ordner localhost/secure
<Location /secure>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession 1
  require shib-session
</Location>

#schützt die Datei 123.txt
<Location /123.txt>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession 1
  require shib-session
</Location>

#Umleitung von https://ulbp2735.ulb.uni-jena.de/dev-mir
#zum Servlet-Container
ProxyPass /dev-mir ajp://ulbp2735.ulb.uni-jena.de:8009/dev-mir

#schützt das Servlet vor unerlaubten Zugriffen
<Location /dev-mir/servlets/MCRShibbolethLoginServlet>
```

```
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

3.4 Testen der Anwendung mit MIR

```
Anwendungen bauen
1. MyCoRe|MIR updaten: svn update
2. MyCoRe bauen: mvn clean install -Denv=dev -DskipTests=true
   oder
   mvn clean install -pl .,mycore-user2
3. MIR bauen: mvn clean install -am -pl mir-webapp

Tomcat extern verwenden
4a. WAR in webapps-Verzeichnis verlinken:
    ln -sfn ../mir-webapp/target/mir-02-Snapshot.war dev-mir
5a. Starten per ./catalina.sh run
6a. Aufrufen über localhost:8080/dev-mir/content/index.xml
7a. bei Änderungen das Verzeichnis '/dev-mir/' in webapps löschen,
    da die WAR-Dateien nur einmal entpackt werden

Tomcat intern verwenden
4b. mvn -Pdev -Dtomcat=8 org.codehaus.cargo:cargo-maven2-plugin:run
    -pl mir-webapp
5b. Aufrufen über localhost:8080/dev-mir/content/index.xml

Jetty intern verwenden
4.c mvn -Pdev -Djetty org.codehaus.cargo:cargo-maven2-plugin:run
    -pl mir-webapp
5.c Aufruf über localhost:8921/dev-mir/content/index.xml
```