Shibboleth

- Shib2IdPMobileLogin
- Shib2IdpSUSE
- Shib3IdpARP
- Shib2IdpTomcat6
- WebmailShibboleth
- Shibboleth2 SP
- Shibboleth2 DiscoveryService
- Shibboleth3 IdP
- Shibboleth Service Provider SP
- Shib2IdpAuth
- ShibAndEdugain
- Shib3IdpMetadata
- Shib3IdpInstall
- Shibenv
- ShibIdPAttrib
- Shibboleth2 IdP
- Shib2SPSourceInstall
- Shib2PersistentId
- Shib2IdpInstall
- Shib2SPConfig
- ShibSPInstallDebian
- Shibboleth Service Provider SP és Docker
- Shibboleth
- Shib2IdpConnectionPool
- ShibIdPX509LdapAuthentication
- Shibboleth SP
- Shib2IdpCluster

Shib2IdPMobileLogin

A Shibboleth autentikációs képernyője tetszőlegesen testre szabható. A fájl a `war/idp.war`-ban található. Módosításhoz ki kell csomagolni, módosítani a `login.jsp`-t majd visszacsomagolni.

Lévén egyre többen használunk webes szolgáltatásokat mobilon keresztül, jó szolgálatot tehet, ha a Shibboleth belépő oldala is optimalizált a kis kijelzőkre és nem kell a nagy fehérségben matatnunk, hogy rámutassunk a felhasználónév mezőre...

Az alábbi, Shibboleth 2.3-hoz készült, módosított login.jsp a jQuery mobile környezetét használja a megjelenítéshez, így a legtöbb mobilon ugyanúgy kell megjelennie. A kód természetesen tovább szabható, formálható. Fontos, hogy javascript nélkül is működik, csak nem lesz szép.

```
<%@ taglib uri="urn:mace:shibboleth:2.0:idp:ui" prefix="idpui" %>
<%

String ua=request.getHeader("User-Agent").toLowerCase();
if(ua.matches(".*(android|avantgo|blackberry|blazer|compal|elaine|fennec|hiptop|iemobile|ip(ho
ne|od)|iris|kindle|lge |maemo|midp|mmp|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\\/|plucker|pocket|psp|symbian|treo|up\\.(browser|link)|vodafone|wap|windo
ws (ce|phone)|xda|xiino).*")||ua.substring(0,4).matches("1207|6310|6590|3gso|4thp|50[1-
6]i|770s|802s|a wa|abac|ac(er|oo|s\\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\\-m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\\-(n|u)|c55\\/|capi|ccwa|cdm\\-
|cell|chtm|cldc|cmd\\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\\-
s|devi|dica|dmob|do(c|p)o|ds(12|\\-d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-
7]0|os|wa|ze)|fetc|fly(\\-|_)|g1 u|g560|gene|gf\\-5|g\\-
mo|go(\\.w|od)|gr(ad|un)|haie|hcit|hd\\-(m|p|t)|hei\\-|hi(pt|ta)|hp( i|ip)|hs\\-c|ht(c(\\-|
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\\-(20|go|ma)|i230|iac( |\\-
|\\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt(
|\\/)|klon|kpt |kwc\\-|kyo(c|k)|le(no|xi)|lg( g|\\/(k|l|u)|50|54|e\\-|e\\/|\\-[a-
w])|libw|lynx|m1\\-w|m3ga|m50\\/|ma(te|ui|xo)|mc(01|21|ca)|m\\-
cr|me(di|rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\\-| |o|v)|zz)|mt(50|p1|v
)|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\\-([1-
8]|c))|phil|pire|pl(ay|uc)|pn\\-2|po(ck|rt|se)|prox|psio|pt\\-g|qa\\-a|qc(07|12|21|32|60|\\-
[2-7]|i\\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\\-|oo|p\\-
)|sdk\\/|se(c(\\-|0|1)|47|mc|nd|ri)|sgh\\-|shar|sie(\\-|m)|sk\\-
```

```
0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\\-|v\\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\\-|tdg\\-|tel(i|m)|tim\\-|t\\-
mo|to(pl|sh)|ts(70|m\\-|m3|m5)|tx\\-9|up(\\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-
3]|\\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|xda(\\-|2|g)|yas\\-|your|zeto|zte\\-")) {
%>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, minimum-scale=1, maximum-scale=1">
        <title>mobil login page</title>
        <link rel="stylesheet"  href="http://code.jquery.com/mobile/1.0a4.1/jquery.mobile-
1.0a4.1.min.css" />
        <script type="text/javascript" src="http://code.jquery.com/jquery-
1.5.min.js"></script>
        <script type="text/javascript"
src="http://code.jquery.com/mobile/1.0a4.1/jquery.mobile-1.0a4.1.min.js"></script>
        <style>
            .errMsg {text-align: center; color: red;}
        </style>
    </head>
    <body>
        <div data-role="page" data-theme="b" id="editor">
            <div data-role="header" data-position="inline">
                <h1>NIIF IdP - mobile login</h1>
            </div>
            <div data-role="content">

                        <% if ("true".equals(request.getAttribute("loginFailed"))) { %>
                            <div data-role="fieldcontain">
                                <p class="errMsg">Authentication Failed</p>
                            </div>
                        <% } %>

                        <% if(request.getAttribute("actionUrl") != null){ %>
                            <form action="<%=request.getAttribute("actionUrl")%>"
method="post" data-ajax="false">
                        <% }else{ %>
```

```
                            <form action="j_security_check" method="post">
                        <% } %>
                            <div data-role="fieldcontain">
                                <label for="j_username">Username:</label>
                                <input type="text" name="j_username" id="j_username" value=""
/>
                            </div>
                            <div data-role="fieldcontain">
                                <label for="j_password">Password:</label>
                                <input type="password" name="j_password" id="j_password"
value="" />
                            </div>
                            <div data-role="fieldcontain" style="text-align:center">
                                <input type="submit" value="Login" data-role="button" data-
inline="true" data-theme="b" />
                            </div>
                        </form>
                </div>
                </div>
        </body>

</html>

<%
} else {

%>

<!DOCTYPE html>
<html>
  <link rel="stylesheet" type="text/css" href="<%= request.getContextPath()%>/login.css"/>
  <head>
    <title>Shibboleth Identity Provider - Example Login Page</title>
  </head>

  <body id="homepage">
    <img src="<%= request.getContextPath()%>/images/logo.jpg" alt="Shibboleth Logo"/>
    <h1>Example Login Page</h1>
    <p>This login page is an example and should be customized.  Refer to the
       <a href="https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthUserPassLoginPage"
```

```
target="_blank"> documentation</a>.
    </p>

    <div class="loginbox">
       <div class="leftpane">
         <div class="content">
           <p>The web site described to the right has asked you to log in and you have chosen
&lt;FILL IN YOUR SITE&gt; as your home institution.</p>
           <% if ("true".equals(request.getAttribute("loginFailed"))) { %>
              <p><font color="red"> Credentials not recognized. </font> </p>
           <% } %>
           <% if(request.getAttribute("actionUrl") != null){ %>
             <form action="<%=request.getAttribute("actionUrl")%>" method="post">
           <% }else{ %>
             <form action="j_security_check" method="post">
           <% } %>
           <table>
             <tr><td width="40%"><label for="username">Username:</label></td><td><input
name="j_username" type="text" /></td></tr>
             <tr><td><label for="password">Password:</label></td><td><input name="j_password"
type="password" /></td></tr>
             <tr><td></td><td><button type="submit" value="Login" >Continue</button></td></tr>
           </table></form>
         </div>
       </div>
       <div class="rightpane">
         <div class="content">
           <div id="spName"><idpui:serviceName/></div>
           <!-- pick the logo.  If its between 64 & max width/height display it
                If its too high but OK wide clip by height
                If its too wide clip by width.
                We should not clip by height and width since that skews the image.  Too high
an image will just show the top.
            -->
           <idpui:serviceLogo  minWidth="64" minHeight="64" maxWidth="350" maxHeight="147"
cssId="splogo">
             <idpui:serviceLogo  minWidth="64" minHeight="64" maxWidth="350"
cssId="clippedsplogoY">
                 <idpui:serviceLogo  minWidth="64" minHeight="64" cssId="clippedsplogoX"/>
             </idpui:serviceLogo>
```

```
            </idpui:serviceLogo>
            <div id="spDescription">
              <idpui:serviceDescription>You have asked to login to
<idpui:serviceName/></idpui:serviceDescription>
            </div>
          </div>
      </div>
    </div>
  </body>
</html>
<% } %>
```

Shib2IdpSUSE

Shibboleth 2 IdP függőségeinek beállítása OpenSUSE alatt

Debian telepítési útmutató: Shib2IdpInstall, ezen az oldalon az OpenSUSE alatt történő azon beállításokat részletezzük, amelyek eltérnek a Debianban megszokottól.

Szükséges csomagok

- apache2
- tomcat6
- java-1_6_0-sun

Apache beállítása

```
chkconfig apache2 on
chkconfig tomcat6 on
#alapbol nincs engedelyezve a proxy_ajp modul
a2enmod proxy proxy_ajp
```

További információk: Shib2IdpInstall#apache-beállítás

Tomcat6 beállítása

A `/etc/tomcat6/tomcat6.conf` fájlban találhatóak meg a környezeti beállítások:

```
JAVA_OPTS="-Xms256M -Xmx512M -XX:-DisableExplicitGC"
JAVA_ENDORSED_DIRS="/usr/local/shibboleth-idp/endorsed"
CLASSPATH="/usr/share/java/mysql-connector-java.jar"
SECURITY_MANAGER="false"
```

További információk: Shib2IdpInstall#Tomcat_6

Shib3IdpARP

Shibboleth 3 IdP attribútum szűrés beállítása

Vonatkozó állományok:

- `{idp.home}/conf/attribute-filter.xml`
- `{idp.home}/conf/idp.properties`

Az `{idp.home}/conf/attribute-filter.xml` állomány már tartalmaz néhány egyszerű példa beállítást. Az alábbi módosítások lehetővé teszik, hogy az attribútum feloldó által feloldott *sn* és *givenName* attribútumok az SP-k számára elérhetők legyenek. Bővítsük az állományt az alábbiak szerint.

```
<afp:AttributeFilterPolicyGroup id="ShibbolethFilterPolicy"
        xmlns:afp="urn:mace:shibboleth:2.0:afp"
        xmlns:basic="urn:mace:shibboleth:2.0:afp:mf:basic"
        xmlns:saml="urn:mace:shibboleth:2.0:afp:mf:saml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:mace:shibboleth:2.0:afp
http://shibboleth.net/schema/idp/shibboleth-afp.xsd
                            urn:mace:shibboleth:2.0:afp:mf:basic
http://shibboleth.net/schema/idp/shibboleth-afp-mf-basic.xsd
                            urn:mace:shibboleth:2.0:afp:mf:saml
http://shibboleth.net/schema/idp/shibboleth-afp-mf-saml.xsd">

<!-- ... további tartalom ... -->

    <!-- ezeket az attribútumokat minden SP megkapja -->
    <afp:AttributeFilterPolicy id="mindensp">
        <afp:PolicyRequirementRule xsi:type="basic:ANY"/>

        <afp:AttributeRule attributeID="sn">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="givenName">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
    </afp:AttributeFilterPolicy>
```

```
    <!-- ezeket az attribútumokat csak a megadott SP kapja meg -->
    <afp:AttributeFilterPolicy id="intezmenysp1">
        <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
value="https://sp1.intezmenyneve.hu/shibboleth"/>

        <afp:AttributeRule attributeID="description">
            <afp:PermitValueRule xsi:type="basic:ANY"/>
        </afp:AttributeRule>
    </afp:AttributeFilterPolicy>

</afp:AttributeFilterPolicyGroup>
```

- A **14.** sor szerint minden SP-re vonatkozik a szabály.
- A **16, 19.** sorban megadott attribútumokat a 14. sor alapján minden SP látja.
- A **26.** sorban megadott SP-re vonatkozóan rendelkezünk.
- A **28.** sor szerint ez az SP megkapja a *description* attribútumot.

Dokumentáció:

- https://wiki.shibboleth.net/confluence/display/IDP30/AttributeFilterConfiguration
- https://wiki.shibboleth.net/confluence/display/IDP30/AttributeFilterPolicyConfiguration

Shib2IdpTomcat6

Ezt a lapot össze kellene vonni ezzel, és az elavult infókat frissíteni

JVM beállítások

A Tomcat6 jelenleg nem működik együtt tökéletesen 6-os JVM-mel (egész pontosan a commons-dbcp csomag a JDBC API pici megváltozása miatt), ezért egyelőre ajánlott 5-ös JVM-mel futtatni - FIXME.

A Shibboleth2 IdP nem hajlandó elindulni a JVM-mel szállított Sun-féle Xerces implementációval, ezért az IdP csomaggal szállított Xerces és Xalan implementációkat be kell másolni a $JAVA_HOME/lib/endorsed könyvtárba, vagy a következő kapcsolóval kell indítani a Tomcat-et:

```
java -Djava.endorsed.dirs=/path/to/xerces-libs
```

Shibboleth IdP telepítése

A kitömörített bináris disztribúció könyvtárában adjuk ki a

```
sh ant.sh
```

parancsot, ami megkérdezi a telepítési könyvtárat (${SHIB_HOME}) és a hostnevet, majd elkészíti azt, legenerálja a teszt-céllal használható kulcsokat és tanúsítványokat (ezeket a credentials könyvtárba teszi idp.key, idp.crt néven).

Ezután a tomcat-et futtató felhasználónak (nálam: tomcat) írási jogot kell adni a log könyvtárra, majd telepíthetjük az idp webalkalmazást a tomcat webapps root alá (nálam: a /var/lib/tomcat-6/webapps/ROOT)

```
chown tomcat:tomcat ${SHIB_HOME}/logs
cp ${SHIB_HOME}/war/idp.war /var/lib/tomcat-6/webapps/ROOT
```

Új SAML SP felvétele

Egy új SP felvételéhez csak az SP metaadatára van szükségünk SAMLv2 szabványos XML formában. A metaadatot vagy fizikailag el kell helyezni a ${SHIB_HOME}/metadata könyvtárban, vagy egy URL-en elérhetővé kell tenni a Shibboleth IdP számára.

Metaadat-források megadása a ${SHIB_HOME}/conf/relying-party.xml -ben

```
<!-- több provider megadása láncolással -->
<MetadataProvider id="ShibbolethMetadata" xsi:type="ChainingMetadataProvider"
xmlns="urn:mace:shibboleth:2.0:metadata">
   <!-- Fájlrendszerből olvasott metaadat -->
   <!-- Fill in metadataFile attribute with deployment specific information -->
   <MetadataProvider id="sp1" xsi:type="FilesystemMetadataProvider"
xmlns="urn:mace:shibboleth:2.0:metadata"
      metadataFile="${SHIB_HOME}/metadata/sp1-meta.xml" maintainExpiredMetadata="true">
    <!--Metaadat aláírás ellenőrzése -->
    <!--MetadataFilter xsi:type="SignatureValidation"
trustEngineRef="shibboleth.MetadataTrustEngine" /-->
   </MetadataProvider>
</MetadataProvider>
```

SAMLv2 Profilok beállítása

A ${SHIB_HOME}/conf/relying-party.xml -ben kell a következő módosításokat eszközölni:

Először a SAMLv2 SSO Profil alapbeállításai

```
<!-- a saját IDP entityID-je, amivel minden SP-hez egyszerre beállíthatjuk mint provider -->
<DefaultRelyingParty
  provider="https://idp.example.com/idp/shibboleth"
  defaultSigningCredentialRef="IdPCredential">

 <!--
   5 perces assertion érvényesség (óraszinkronizálás IdP - SP között fontos!)
   A válaszok kötelező digitális aláírása
   Attribútum push
 -->
 <ProfileConfiguration xsi:type="saml:SAML2SSOProfile"
    includeAttributeStatement="true"
    assertionLifetime="300000"
```

```
    assertionProxyCount="0"
    signResponses="always"
    signAssertions="never"
    encryptAssertions="conditional"
    encryptNameIds="conditional" />
</DefaultRelyingParty>
```

SP esetén az alapértelmezett beállítások felülírása:

```
<RelyingParty
    id="https://sp1.example.com/shibboleth"
    provider="https://idp.example.com/idp/shibboleth"
    defaultSigningCredentialRef="IdPCredential">
   <ProfileConfiguration xsi:type="saml:SAML2SSOProfile" encryptAssertions="never"/>
</RelyingParty>
```

Attribútum kiadása (címtárból)

Egy attribútum kiadásához két dolgot kell beállítani: a resolver-t és a filter-t. Előbbi felelős az attribútum megszerzéséért és a session kontextusba helyezésért, utóbbi az SP felé történő kiadást szabályozza.

Új attribútum beolvasása címtárból (${SHIB_HOME}/conf/attribute-resolver.xml) és SAMLv1 illetve SAMLv2 AttributeStatement -be kódolása:

```
<resolver:AttributeDefinition id="email" xsi:type="Simple"
  xmlns="urn:mace:shibboleth:2.0:resolver:ad"
  sourceAttributeID="mail">
  <resolver:Dependency ref="myLDAP" />
  <resolver:AttributeEncoder xsi:type="SAML1String"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:mace:dir:attribute-def:mail" />
  <resolver:AttributeEncoder xsi:type="SAML2String"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail" />
</resolver:AttributeDefinition>
```

A resolver:Dependency adja meg azt a forrást, amiből az attribútum feloldásra kerül. Ez esetünkben a myLDAP:

```
<resolver:DataConnector id="myLDAP" xsi:type="LDAPDirectory"
   xmlns="urn:mace:shibboleth:2.0:resolver:dc"
   ldapURL="ldap://ldap.example.com" baseDN="ou=people,dc=example,dc=com"
   principal="userid=shibboleth,ou=systems,dc=example,dc=com"
   principalCredential="password">
     <FilterTemplate>
           <![CDATA[
               (uid=$requestContext.principalName)
           ]]>
     </FilterTemplate>
</resolver:DataConnector>
```

NameIdentifier leképzés

Szintén az attribute-resolver.xml -ben kell beállítani az Assertion Subject NameID -t, ami az IdP-SP közötti azonosítóért felel. Példaképp egy SAMLv2 Tranziens azonosítót a következőképp állíthatunk be:

```
<resolver:AttributeDefinition id="transientId" xsi:type="TransientId"
xmlns="urn:mace:shibboleth:2.0:resolver:ad">
  <resolver:AttributeEncoder xsi:type="SAML2StringNameID"
    xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
    nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />
</resolver:AttributeDefinition>
<resolver:PrincipalConnector xsi:type="Transient"
  xmlns="urn:mace:shibboleth:2.0:resolver:pc"
  id="saml2Transient"
  nameIDFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />
```

WebmailShibboleth

Shibboleth, Webmail, IMAP Proof-of-concept

In English

Requirements

- The webmail software must not see or use users' LDAP password, the IdP must not release even the hashed form of the password.
- IMAP must authenticate with username and password.
- If one has access to the webmail server, she must not have access to the IMAP on behalf of all users (she can however access to active users session).

Solution concepts

- The IdP and the IMAP server share an authentication database.
- With every webmail SP request the IdP generates a new password for that particular user and writes it to the database.
- The webmail SP receives this password with the attribute set and uses the username (e-mail address) and password to access the IMAP server.
- The IMAP server tries to authenticate against the database.
- In order to secure access, this password entry should contain an expiration time, which invalidates the password after the IdP session ends, so IMAP accepts only those users who has recently initiated active session at the IdP side.

Shibboleth IdP plugin

- We have developed an IdP plugin -attribute resolver- which can generate this short-lifetime password (called service token) for the user and write it to the database.

- Shibboleth IdP attribute resolver configuration is independent from the actual SP, so the plugin must check whether the current request came from an SP for which it needs to generate the token.
- The service token is sent in plain-text, so the Shibboleth attribute statement must be encrypted either by using artifact resolution over SSL/TLS or by using XML encryption with HTTP-Post.

IMAP configuration

- As we don't want to force the use of webmail, IMAP needs to use LDAP authentication as well.
- Most IMAP servers can be configured to use PAM, which can be configured to use arbitrary SQL tables for authentication and it also supports authentication chaining.

Webmail softwares

- For our proof-of-concept we have tried squirrelmail and roundcube with its HTTP-authentication plugin. If the SP is releasing the username and service token as PHP_AUTH_USER and PHP_AUTH_PW, this authentication module works out-of-the-box.

Magyarul

Koncepció

A webmail és a levelezőszerver (IMAP/POP3) együttes működését szeretnénk Shibbolizálni. A fő probléma abból áll, hogy a webmail az IMAP szerver felé felhasználónévvel és jelszóval autentikál. Az címtárban tárolt jelszót azonban nem adhatjuk ki az alkalmazásoknak, ráadásul legtöbb esetben ez egy hashelt jelszó.

A következő kritériumoknak kell teljesülniük:

- a webmail nem fér hozzá a felhasználó SSO jelszavához (még hashelt formátumban sem)
- az IMAP szerver jelszavas autentikációt használ, minden felhasználónak egyedi jelszava van
- a webmail feltörése esetén nem férhetnek hozzá az összes felhasználó levelezéséhez

A fenti kritériumokat az 'egyszer használatos', rövid lejáratú jelszó használata ('service token') kielégíti. Ebben az esetben az IdP minden egyes webmail bejelentkezéshez generál egy véletlen jelszót, és ezt elmenti egy adatbázisban, (beállítva a jelszóhoz egy rövid lejárati időt) valamint elküldi a webmail SP-nek. A webmail ezen rövid lejáratú jelszó használatával autentikál az IMAP

szerver felé.

A leírt gondolatmenet megvalósításához három komponens együttműködése szükséges:

- az IdP jelszót kell generáljon egy adatbázisba
- a webmailnek el kell érnie ezt a jelszót
- az IMAP szervernek a jelszóadatbázist kell használnia az autentikációra

Adatbázis struktúra

MySQL használata esetén a következő adatbázisstruktúra használható:

```
CREATE TABLE `service_tokens` (
  `uid` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `expiration` datetime NOT NULL,
  PRIMARY KEY  (`uid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

IdP plugin

Az IdP plugin aktuális verziója a következő URL-ről tölthető le:

http://software.niif.hu/maven2/hu/niif/shibboleth-servicetoken/1.0. A `shibboleth-servicetoken-1.0.jar` -t illetve a megfelelő adatbázis drivert (MySQL esetén `mysql-connector.jar`) be kell másolni az `idp.war WEB-INF/lib` könyvtárába.

Az `attribute-resolver.xml` -ben a következő változtatásokat kell megtenni:

```
<!--xml semak megfelelo beallitasa -->
<AttributeResolver
  ....
  xmlns:niifconnector="urn:geant:niif.hu:dataconnector"
  xsi:schemaLocation="
      ....
      urn:geant:niif.hu:dataconnector classpath:/schema/servicetokendataconnector.xsd">

 <!-- onetimepassword definicio -->
 <resolver:AttributeDefinition id="serviceToken" xsi:type="Simple"
   xmlns="urn:mace:shibboleth:2.0:resolver:ad"
   sourceAttributeID="serviceToken">
```

```
    <resolver:Dependency ref="serviceTokenConnector" />

    <resolver:AttributeEncoder xsi:type="SAML2String"
        xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:geant:niif.hu:servicetoken" friendlyName="serviceToken" />
  </resolver:AttributeDefinition>

  <!-- uid definicio -->
  <resolver:AttributeDefinition id="uid" xsi:type="Simple"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
     sourceAttributeID="uid">
    <resolver:Dependency ref="myLDAP" />
    <resolver:AttributeEncoder xsi:type="SAML1String"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:mace:dir:attribute-def:uid" />
    <resolver:AttributeEncoder xsi:type="SAML2String"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid" />
  </resolver:AttributeDefinition>

  <!-- service token generalasa -->
  <resolver:DataConnector xsi:type="niifconnector:ServiceToken"
    id="serviceTokenConnector"
    sourceAttributeID="uid"
    generatedAttributeID="serviceToken"
    tableName="service_tokens"
    principalColumn="uid"
    passwordColumn="password"
    expirationColumn="expiration"
    passwordLifetime="XXXXXX"
    spEntityID="https://webmail.example.org/shibboleth" >

    <resolver:Dependency ref="myLDAP" />

    <dc:ApplicationManagedConnection
        jdbcDriver="com.mysql.jdbc.Driver"
        jdbcURL="jdbc:mysql://localhost:3306/shib_idp"
        jdbcUserName="*****"
        jdbcPassword="*****" />
```

```
  </resolver:DataConnector>
```

Fontos, hogy a `DataConnector` (másodpercekben értelmezett) `passwordLifetime` attribútumát jól állítsuk be, azaz hosszabb legyen, mint a webmail oldali SP session, de javasolt 24 óránál rövidebbre venni.

Az `attribute-filter.xml` -ben pedig ki kell engedni az `uid` és `serviceToken` attribútumokat a webmail sp-nek:

```
<AttributeFilterPolicy id="sendServiceTokenToWebmail">
 <PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
  value="https://webmail.example.org/shibboleth" />

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

IMAP konfiguráció (Cyrus imapd)

Imapd saját autentikáció

A Cyrus imapd-ben be kell állítani az SQL autentikációt. Ehhez a libsasl2-modules-sql debian csomagra is szükségünk lesz. A konfigurációt az imapd.conf-ban tehetjük meg:

```
sasl_mech_list: PLAIN
sasl_pwcheck_method: auxprop

sasl_auxprop_plugin: sql
sasl_sql_engine: mysql
sasl_sql_hostnames: localhost
sasl_sql_user: *****
sasl_sql_passwd: *****
sasl_sql_database: shib_idp
sasl_sql_select: SELECT password AS userPassword FROM service_tokens WHERE uid = '%u' AND
```

```
expiration > now()
```

Amennyiben az IMAP szervert nem TLS/SSL felett használjuk, ezek a beállítások nem biztonságosak!

Használat PAM-mal

PAM használata esetén távolítsuk el a libsasl2-modules-sql csomagot, mert felesleges logüzeneteket gyárt. Ezen kívül szükség van a saslauthd-re is, amit debian alatt a sasl2-bin csomagban találhatunk.

Az imapd.conf-ot a következőképp kell beállítani:

```
sasl_mech_list: PLAIN
sasl_pwcheck_method: saslauthd
```

A saslauthd-t az /etc/default/saslauthd fájlban kell engedélyeznünk:

```
START=yes
MECHANISMS="pam"
```

Az /etc/pam.d/imap fájlban kell az imap pam beállításokat megtenni. Adatbázis használatához a libpam-mysql csomag is szükséges. Ha az adatbázisos felhasználókhoz nincs lokális account, akkor a PAM 'account' metódusát permit-re kell állítani.

```
auth       sufficient   pam_ldap.so
auth       sufficient   pam_mysql.so use_first_pass user=****** passwd=****** \
  host=/var/run/mysqld/mysqld.sock db=shib_idp table=service_tokens usercolumn=uid
passwdcolumn=password \
  crypt=plain [where=expiration>now()]
auth       required     pam_deny.so
@include common-account
```

SP konfiguráció

A webmailt futtató webszerver Shibboleth konfigurációjában el kell fogadni a felhasználónevet és a jelszót az IdP-től. Az attribute-map.xml -hez a következő bejegyezéseket kell hozzáadni:

```
  <Attribute name="urn:oid:0.9.2342.19200300.100.1.1" id="PHP_AUTH_USER"/>
  <Attribute name="urn:geant:niif.hu:servicetoken" id="PHP_AUTH_PW"/>
```

Figyeljünk arra, hogy az attribute-policy.xml -ben ezeket az attribútumokat beengedjük! (Az alapértelmezett telepítés egy catch-all engedélyező szabályt tartalmaz, tehát az attribútum rendben meg fog jelenni.)

Webmail szoftverek konfigurációja

Squirrelmail

A Squirrelmailhez a Squirrelmail HTTP Authentication Plugin letöltésével és telepítésével elvégezhető az IdP által kiadott és az SP által láthatóvá tett felhasználónév és jelszó alapú bejelentkezés.

Roundcube

A HTTP Authentication Plugin telepítése után a plugin-ból el kell távolítani a következő sort:

```
public $task = 'login';
```

Shibboleth2 SP

Telepítési leírások

- Shibboleth 2 SP telepítése forrásból

Konfigurációs leírások

- Shibboleth 2 SP konfigurálása

Shibboleth2 DiscoveryService

Shibboleth2 Discovery Service

A Discovery Service ugyanazt a szerepet játssza a Shibboleth2 infrastruktúrában mint a WAYF szolgáltatás a Shibboleth1.x esetén. A WAYF szolgáltatás azonban Shibboleth-specifikus, míg a Discovery Service pontos működését a SAML2.0 szabvány írja le.

Részletes konfigurációs segédlet a Shibboleth2 Wikin

Az IdP Discovery menete

A Discovery Service (DS) feladata tehát, hogy az SP számára kiválasszon egy IdP-t, amit a felhasználó használ. Ezt alapvetően kétféleképpen éri el: cookie-k illetve egy webes IdP kiválasztó felület segítségével.

Amennyiben a DS cookie-ban tárolja a felhasználó által kiválasztott IdP-t, ezt egy _saml_idp nevű cookieban kell tennie, aminek a formátuma kötött: egy darab space-szel elválasztott, base64 kódolt IdP entityID lista (tehát több IdP-t is képes tárolni, és a használati sorrendet is megőrzi - bár utóbbit nem teszi kötelezővé a szabvány).

A Shibboleth2 DS képes arra, hogy a cookie-k használatán kívül a felhasználónak egy webes formot mutasson, amiben a federációk és az egyes federációkban szereplő IdP-k listáját jeleníti meg. Ezt a listát a felhasználó egy szöveges kereső segítségével szűrheti is.

A Discover Service metaadatok konfigurálása

A DS konfigurálása a wayfconfig.xml fájl segítségével történik. Itt kell megadni a használt template-et és a metaadatokat is. A metaadatokat ugyanazon formátumban kell konfigurálni mint az IdP esetén (tehát lehetőség van egy URL-ről automatikusan letöltött metaadat-állomány használatára is).

A DS-nek ismernie kell a kérő SP metaadatát is, alapbeállításként pedig csak azokat az IdP-ket listázza, amelyek egy metaadat-fájlban vannak az aktuális SP-vel.

Discovery Service pluginek

A DS kiterjeszthető olyan pluginekkel, amelyek segítik a felhasználót az IdP választásban. Ilyen plugin például a _saml_idp cookie kezelő, ami SAML2 szabvány-kompatibilissé teszi a DS-t.

Shibboleth3 IdP

Telepítési leírások

- Shibboleth 3 IdP telepítése (Debian 8 + Jetty 9.2)
- Shibboleth 3 IdP éles szolgáltatás építése (unix service)

Konfigurációs leírások

- Shibboleth 3 IdP metaadat beállítása
- Shibboleth 3 IdP hitelesítés beállítása
- Shibboleth 3 IdP attribútum feloldás beállítása
- Shibboleth 3 IdP attribútum szűrés beállítása

Shibboleth Service Provider SP

Az alábbi lapon megkíséreljük összefoglalni a legfontosabb lépéseket, melyek általános esetben elegendőek ahhoz, hogy működő Shibboleth SP-t állítsunk üzembe. Fontos, hogy rengeteg olyan igény lehet, amely további speciális beállítások meglétét teszik szükségessé, ezeket ezen a lapon nem részletezzük, ilyen irányú tájékozódáshoz legbiztosabb forrás a http://wiki.shibboleth.net lap.

Telepítés

Előkészületek

A Shibboleth SP egy webszerver modul, így szükséges előfeltétel, hogy a gépünkön legyen egy webszerver telepítve, jelenleg Apache és IIS a biztosan támogatott webszerverek. Emellett szükséges, hogy a 443-as port a tűzfalon mindkét irányban engedélyezett legyen, ill. a hosztnév, amelyen a webszerver üzemel, be legyen jegyezve a DNS-be.

Letöltés és telepítés

A legfrissebb változat letölthető https://wiki.shibboleth.net/confluence/display/SP3 címről. Célszerű valamilyen előre csomagolt változattal dolgozni, melyet az adott rendszer csomagkezelőjén keresztül egy mozdulattal feltelepíthetünk minden függőségével együtt.

- Az alábbi parancs a Shibboleth webszerver modul függőségeit is telepíti. Előfordulhat, hogy a telepítés engedélyezni kell a modult és újraindítani az Apache webszervert.

Debian 9 “Stretch” / Ubuntu 16.04 LTS (Xenial)

```
sudo apt install apache2 libapache2-mod-shib2
```

Debian 8 “Stretch” / Ubuntu 14.04 LTS (Trusty)

```
sudo apt-get install apache2 libapache2-mod-shib2
```

Debian 6 “Squeeze” / Debian 7 “Wheezy” / Ubuntu 12.04 LTS (Precise)

Debian 6, Debian 7 és Ubuntu 12.04 rendszerek támogatása lejárt, így a Shibboleth csomag telepítését sem ajánljuk ezekre a rendszerekre!

A Debian Shibboleth csapat által készített új verziók időről időre bekerülnek backports.org tárolóiba is, ezért stable Debiant futtató rendszereken javasolt ezt használni.

A backports.org tárolójának beállításához adjuk hozzá a `/etc/apt/sources.list` fájlhoz a következő sort:

```
deb http://backports.debian.org/debian-backports squeeze-backports main
```

A csomagokat így telepíthetjük:

```
sudo aptitude update
sudo aptitude install -t squeeze-backports libapache2-mod-shib2
```

Ez a Shibboleth webszerver modul függőségeit is telepíti. A Shibboleth használatba vételéhez engedélyezni kell a modult és újra kell indítani az Apache webszervert.

Red Hat / CentOS (RPM) alapú disztribúciók

Shibboleth SP-ből RHEL/CENTOS rendszerekre egyből rendelkezésünkre áll bináris csomag, a fejlesztők ezen platformokon dolgoznak első körben. A telepítés előtt a teendőnk mindössze annyi, hogy a YUM forrásokhoz hozzá kell adni a Shibboleth SP repóját. Ehhez látogassunk el a http://download.opensuse.org/repositories/security://shibboleth/ oldalra, majd a pontos verzió kiválasztása után a security:shibboleth.repo fájl tartalmát másoljuk be a /etc/yum/ `/etc/yum.repos.d/CentOS-Base.repo` fájl aljára, majd

```
yum install shibboleth
```

Hibaelhárítás: Can't connect to listener process

Ha a fenti hibába futunk bele, az azt jelenti, hogy a SE linux nem engedi kommunikálni a shibd és a httpd folyamatokat, ezért ezt külön engedélyezni kell. Ehhez készítsünk egy fájlt `shibd.te` néven, melynek tartalma az alábbi legyen:

```
module shibd 1.0;
require {
        type var_run_t;
        type httpd_t;
```

```
        type initrc_t;
        class sock_file write;
        class unix_stream_socket connectto;
}
####### # httpd_t ######=
allow httpd_t initrc_t:unix_stream_socket connectto;
allow httpd_t var_run_t:sock_file write;
```

Ezek után futtassuk le az alábbi parancsokat, melyek a fenti fájlt megfelelően lefordítják, és be is illesztik a létrehozott szabályunkat.

```
# checkmodule -M -m -o shibd.mod shibd.te
checkmodule:  loading policy configuration from shibd.te
checkmodule:  policy configuration loaded
checkmodule:  writing binary representation (version 6) to shibd.mod
# semodule_package -o shibd.pp -m shibd.mod
# semodule -i ./shibd.pp
```

Alapbeállítások

A telepített Shibboleth SP konfigurációs állományait Linux alatt a /etc/shibboleth könyvtárban találjuk. Itt a `shibboleth2.xml` és az `attribute-map.xml` fájlokkal lesz dolgunk.

`shibboleth2.xml`

A telepítéskor alapértelmezetten érkező fájl nagyrészt megfelelő számunkra, az induláshoz csak az alább részletezett módosításokat kell megejtenünk. A változtatandó kódrészletek mind szerepelnek már az eredeti xml-ben is, így a feladat többnyire csak változtatásról, átírásról, bizonyos részek kommentjelek közül történő kiszabadításáról szól.

- Választanunk kell egy entityID-t. Ez általában a védendő szolgáltatást futtató hosztnévből származik: `https://hosztnev/shibboleth`, ezt az azonosítót beírni az `ApplicationDefaults` részbe az alábbi módon (az entityID és a homeURL értékein kívül, ahová a hosztnév írandó be, alapértelmezés szerint mást nem szükséges változtatni):

  ```
  <ApplicationDefaults entityID="https://hosztnév/shibboleth"
                       homeURL="https://hosztnév/shib-error.html"
                       signing="false" encryption="false"
                       id="default" policyId="default"
                       REMOTE_USER="eppn persistent-id targeted-id">
  ```

- Meg kell adni, hogy egy Discovery Service-n keresztül kérjük meg a felhasználót, hogy adja meg, mely IdP-től érkezik, avagy csak IdP-t engedélyezünk számukra. Az első eset nyilvános szolgáltatások esetében indokolt, a második belső, pl. csak intézményi oldalak védésénél szükséges.
  - Discovery Service beállítása

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://discovery.eduid.hu">
SAML2 SAML1
</SSO>
```

  - Fix IdP beállítása

```
<SSO entityID="https://idp.example.org/idp/shibboleth">
SAML2 SAML1
</SSO>
```

- Meg kell adnunk, hogy milyen metadata forrásból dolgozzon az SP, az alábbi példában az eduID-ban használatos beállítás látható (a hivatkozott href-metadata-signer-2011.crt fájl a http://metadata.eduid.hu címről töltendő le a shibboleth konfigurációs könyvtárába) :

```
<MetadataProvider type="XML" uri="http://metadata.eduid.hu/current/href.xml"
    backingFilePath="href.xml" reloadInterval="7200">
    <MetadataFilter type="Signature" certificate="href-metadata-signer-2011.crt"/>
</MetadataProvider>
```

- Meg kell adni, hogy az SP mely kulcsot és tanúsítványt használja. Ehhez egy self-signed tanúsítványra lesz szükség, amely pl. az alábbi paranccsal generálható:

```
  openssl req -new -newkey rsa:2048 -x509 -days 3652 -nodes -out sp.example.org.crt -
keyout sp.example.org.key
```

  Az xml-ben módosítandó részlet pedig:

```
  <CredentialResolver type="File" key="sp.example.org.key"
certificate="sp.example.org.crt"/>
```

- Végül ugorjunk vissza a fájl első felére, szedjük ki a kommentjeleket a `RequestMapper` rész körül, adjuk meg a kezelendő hosztokat és path-okat. Egy Shibboleth SP példány több, az adott webszerver által kezelt hosztot is kezelni tud, és egy-egy hoszton belül több útvonalat is, ezeket itt meg kell adni. Alább egy példa:

```
<RequestMapper type="Native">
    <RequestMap applicationId="default">
        <Host name="hosztnév" authType="shibboleth" requireSession="false">
                <Path name="secure" authType="shibboleth" requireSession="true" />
```

```
        </Host>
    </RequestMap>
</RequestMapper>
```

A péda szerint a hosztnév alatti tartalom nem kíván meg shibbolethes azonosítást, kivéve a `/secure` location. Ha valahol shibbolethes azonosítást kívánunk használni, azt ezen kívül az adott hoszt webszerver konfigjában is jeleznünk kell. Apache-nál az alábbi módon.

```
<Location /secure>
 AuthType shibboleth
 require valid-user
 ShibUseHeaders On
 ShibRequireSession On
</Location>
```

További részletek az authorizációval kapcsolatban:

https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPhtaccess

SP adatainak közzététele

A fenti beállítások után egy működő Shibboleth SP-t kapunk eredményül, ugyanakkor ténylegesen csak akkor fog tudni együttműködni a föderációban résztvevő IdP-kkel, ha az SP metaadatait közzétesszük, így azt az IdPk megismerhetik. Ehhez az eduID föderációban a frissen beállított SP adatait a Resource Registry nevű webes adminisztrációs oldalon kell felvinni egy varázsló segítségével, majd a jóváhagyás után várni pár órát, amíg a változások érvénybe lépnek. Ha nincs az intézményi entitások menedzseléséhez jogod ('Access denied'), akkor konzultálj az intézményed eduID kapcsolattartójával, hogy az SP-det szeretnéd a föderációba regisztrálni. Segítséget nyújt az alábbi lista: https://rr.eduid.hu/list

HEXAA integráció

Ha az eduID-ban használatos külső attribútum forrást is szeretnénk az SP-nkhez illeszteni, akkor a shibboleth2.xml fileban a következő példán keresztül lehet megtenni.

```
<AttributeResolver type="Chaining">
   <AttributeResolver type="Query"/>
   <AttributeResolver type="SimpleAggregation" attributeId="eppn"
```

```
format="urn:oid:1.3.6.1.4.1.5923.1.1.1.6">
       <Entity>https://hexaa.eduid.hu/hexaa</Entity>
       <Attribute Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
FriendlyName="eduPersonEntitlement"/>
   </AttributeResolver>
 </AttributeResolver>
```

Kapcsolódó lapok

- Shibboleth 2 SP telepítése FastCGI alapú webszerver környezetben

Shib2IdpAuth

Autentikáció nativan, Shib2Idp-ből

LDAP-alapon

Szerkesszük a `${SHIB_HOME}/conf/login.config` -ot:

```
ShibUserPassAuth {
  edu.vt.middleware.ldap.jaas.LdapLoginModule required
     host="ldap.example.com"
     base="ou=people,dc=example,dc=com"
     ssl="false"
     serviceUser="userid=example-system,ou=systems,dc=example,dc=com"
     serviceCredential="password"
     userField="uid";
}
```

A serviceUser és a serviceCredential kihagyható, ekkor anonymous bind történik (azonban ilyen esetben a helytelen név / jelszó megadása LDAP Exception-t okoz és nem a jól értelmezhető hibás név / jelszó üzenetet adja a felhasználónak)

Ezután be kell állítani, hogy ezt a bekonfigurált autentikációt használja a Shibboleth (${SHIB_HOME}/conf/handlers.xml)

```
<LoginHandler xsi:type="UsernamePassword"
                 authenticationDuration="240"
                 jaasConfigurationLocation="file://${SHIB_HOME}/conf/login.config">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</Authe
nticationMethod>
</LoginHandler>
<!-- SSO-hoz kell hogy az előző session-t át tudja venni -->
<LoginHandler xsi:type="PreviousSession">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession</AuthenticationMe
```

```
thod>
</LoginHandler>
```

Az authenticationDuration paraméter elhagyása esetén az IdP 30 perc érvényességgel állítja ki a munkamenetet (`<AuthnStatement SessionNotOnOrAfter="...">`), tehát akár aktív tevékenység esetén is 30 perc múlva lejár a felhasználó session-je. Ezt érdemes tehát átállítani magasabb értékre.

A FORM-ot az idp.war -ban tudjuk testreszabni (login.jsp). A szállított login.jsp által beállított FORM tag és INPUT tag-ek tartalmát ne módosítsuk!

Ha a bejelentkezés nem sikerül jó felhasználónév/jelszó párral sem, a `logging.xml` szerkesztésével tudjuk megjeleníteni a debug üzeneteket (a `edu.vt.middleware.ldap` loggert kell átkonfigurálni).

Kerberos alapon

> **A szócikk vagy fejezet még megírásra vár**
>
> Ha ki tudod egészíteni, megköszönjük!

SQL (JDBC) alapon

A Shibboleth 2 IdP képes az autentikálást szabványos JAAS (Java Authentication and Authorization Service) modulokkal elvégezni, ezért lehetőség van relációs adatbázist használó autentikációs modul használatára is. Számos JAAS modul létezik adatbázisos autentikációra is, azonban ezek vagy túl bonyolultak, vagy nem kellően rugalmasak. Az alábbiakban az NIIF által fejlesztett (a Tagish/JDBC kódbázisán alapuló) modul beállítását mutatjuk be:

- JAAS/JDBC modul megfelelő verziójának letöltése
- jaas-jdbc-VERSION.jar és adatbázis driver jar bemásolása az idp webalkalmazás (idp.war) WEB-INF/lib könyvtárába
  - a MySQL Connector/J letölthető a MySQL oldalról
  - MySQL esetén a mysql-connector-java-{verzio}-bin.jar fájlra van szükségünk
- handler.xml -ben UsernamePassword login handler engedélyezése és RemoteUser login handler tiltása
- login.config ShibUserPassAuth-ban a JDBCLoginModul engedélyezése (a többi JAAS modul legyen kikommentezve!)
- adatbázis kapcsolattal összefüggő beállítások:
  - "hagyományos" megoldás
    - **dbDriver**: JDBC Driver osztály neve
    - **dbURL, dbUser, dbPassword**: adatbázis elérési paraméterek
  - JNDI használata esetén

    - **jndiResourceName**: DataSource API-t támogató JNDI név (bővebben lásd: Connection pool leírás)
- egyéb beállítások
    - **usersPreparedStatement**: egy olyan lekérdezés, ami a tárolt elhashelt jelszót kérdezi le egy felhasználónévhez (a felhasználónév helyén a ? karakter kell álljon, a lekérdezés egy vagy nulla sort kell visszaadjon!)
    - **passwordHashMethod**: a hasheléshez alkalmazott metódus (a használható metódusakat a Java Cryptography Architecture dokumentáció írja le).

A JAAS modul konfigurációja a login.config fájlban:

```
hu.niif.middleware.jaas.JDBCLoginModule required
  dbDriver="com.mysql.jdbc.Driver"
  dbURL="jdbc:mysql://databaseHost:3306/databaseName"
  dbUser="dbuser"
  dbPassword="randomsecret"
  usersPreparedStatement="SELECT password FROM users where username=?"
  passwordHashMethod="MD5";
```

LDAP-ból ellenőrzött X.509 tanúsítvánnyal

Ezen autentikációs mód a konténer (pl. Apache) által a klienstől elkért klienstanúsítványt veti össze a felhasználó LDAP bejegyzésében tárolt tanúsítványokkal (`userCertificate`). A modul használatának előfeltételei:

- a konténernek támogatnia kell a kliens-tanúsítványokat, azonban a CA ellenőrzés nem követelmény, a felhasználók self-signed tanúsítvánnyal is igénybe vehetik az autentikációs szolgáltatást
- az IdP-nek a kérésből el kell érnie a klienstanúsítványt
- a tanúsítványban szerepelnie kell a felhasználónévnek (mégpedig az `UID` mezőben)

A modul dokumentációja a ezen az oldalon érhető el.

Autentikáció konténer által

MySQL Autentikáció Apache-on keresztül

Az alábbiakban leírt Apache beállítások elsőre nyakatekertnek tűnhetnek, de az Apache 2.2-es sorozatában előforduló - ez idáig érdemben nem javított - bug miatt ez a megoldás működik csak.

Telepíteni kell a MySQL autentikációs Apache modult:

```
apt-get install libapache2-mod-auth-mysql
```

Engedélyezni kell a modul használatát

```
a2enmod auth_mysql
```

Az Apache adott hosthoz tartozó configjában meg kell adni:

```
Auth_MySQL_Info <host> <DB_user> <DB_password>
```

Ugyanitt meg kell adni az adott Location-re vonatkozó beállításokat - itt látja majd a webszerver, hogy ha az adott url-re érkezett kérés, akkor MySQL-ből kell autentikálnia az itt megadott paraméterek szerint

```
<Location /whereTo/Authn/RemoteUser>
 AuthType Basic
 AuthName "You can login here"
 AuthUserFile /dev/null
 AuthBasicAuthoritative Off

 AuthMySQL on
 AuthMySQL_Authoritative off
 AuthMySQL_DB VHOtools
 AuthMySQL_Password_Table vho_Users
 AuthMySQL_Password_Field password
 AuthMySQL_Encryption_Types PHP_MD5
 require valid-user
</Location>
```

MySQL Autentikáció TomCat-en keresztül

> **A szócikk vagy fejezet még megírásra vár**
>
> Ha ki tudod egészíteni, megköszönjük!

LDAP Autentikáció Apache-on keresztül

A szócikk vagy fejezet még megírásra vár

Ha ki tudod egészíteni, megköszönjük!

Single Sign-on

IdP Session

Az IdP-ben a session-nek nincs rögzített lifetime-ja, hanem aktivitásfüggő timeout értéket lehet beállítani. A jelenlegi IdP-ben az IdP session timeout független a fent megadott `authenticationDuration` értéktől, ezt az `internal.xml` állományban állíthatjuk be:

```
    <bean id="shibboleth.SessionManager"
          class="edu.internet2.middleware.shibboleth.idp.session.impl.SessionManagerImpl"
          depends-on="shibboleth.LogbackLogging">
        <constructor-arg ref="shibboleth.StorageService" />
        <constructor-arg value="<b>28800000</b>" type="long" />
    </bean>
```

A példában a StorageService konstruktorának értéke *ezredmásodpercben* értendő.

Single Logout

ShibAndEdugain

Loading metadata

Metadata downloaded from https://mds.edugain.org

Strange things

- Metadata is not signed by a third party
- Line breaks and indentation is quite by chance, however running through `xml_pp` of course invalidates the signature of the individual `<EntityDescriptor>`s
- Metadata cannot be validated to the schema (see later)

Problems loading metadata to Shibboleth SP

For perl processing, MDS output is run through `xml_pp`, an XML pretty-printer.

Here is the command I use to load MDS output to a Shibboleth 2.0 SP:

```
wget -O- --ca-certificate=/home/bajnokk/edugain_bundle.crt https://mds.edugain.org |xml_pp \
| perl -pe 's/(<(md:)?EntitiesDescriptor)/\1 xmlns="urn:oasis:names:tc:SAML:2.0:metadata"/;
s/.*RoleDescriptor.*//g; s/.*OnlineCA.*//g; \
           s/cacheDuration[>](^)*//g; ' >/tmp/mds-pp.xml
```

Explanation follows:

Unable to connect

For some reason, Shibboleth 2.0 cannot connect to https://mds.edugain.org. It seems to be a `libcurl` issue, which is not easy to circumvent. (See this shib-users thread) Newer cURL's can handle the SSL handshake (the ones in Ubuntu Intrepid and Debian Lenny can not). So it's necessary to `wget` the metadata.

It turned out that newer versions of Shibboleth can connect to mds.edugain.org, however the following errors prevent the metadata from being loaded directly.

No default namespace

There is no default namespace for the outer `EntitiesDescriptor`, the root element. No problem with that, but there is at least one `EntityDescriptor`, which is not correctly namespaced (and assumes that the default namespace is `urn:oasis:names:tc:SAML:2.0:metadata`)

Solution:

```
| perl -pe 's/(<(md:)?EntitiesDescriptor)/\1 xmlns="urn:oasis:names:tc:SAML:2.0:metadata"/;'
```

Invalid use of RoleDescriptor

SAML Metadata Schema declares that RoleDescriptor is an abstract element, whatever it means. Shibboleth (2.0) cannot load an entity with such an element.

Solution: | perl -pe 's/.*RoleDescriptor.*//g;' At the time of writing, it only affects Fresco-AAI. For some unknown reason, Fresco-AAI metadata is a one-liner (even after pretty printing), so it's possible to remove it such a way. If it wasn't the case, proper XSLT would be necessary.

Invalid extension of the schema

GIdP entity contains an `egmd:OnlineCADescriptor` element, which is not a standard extension of the SAML schema.

Solution:

```
| perl -pe 's/.*OnlineCA.*//g;'
```

At the time of writing, it only affects GIdP. For some unknown reason, GIdP metadata is a one-liner (even after pretty printing), so it's possible to remove it such a way. If it wasn't the case, proper XSLT would be necessary.

Shib3IdpMetadata

Shibboleth 3 IdP metaadat beállítása

Vonatkozó állományok:

- `{idp.home}/conf/metadata-providers.xml`

Nem föderációs SP metaadat

Nem föderációs SP metaadatának felvitele a `metadata-providers.xml` állományba az alábbiak szerint történik. A példában szereplő `%{idp.home}/metadata/sp-sp.intezmenyneve-metadata.xml` állományt el kell helyezni az állományrendszerben és meg kell győződni annak valódiságáról (nincs aláírva).

```
<MetadataProvider id="ShibbolethMetadata" xsi:type="ChainingMetadataProvider" ... >

<!-- ... további tartalom ... -->

    <!-- Helyi, nem föderációs SP -->
    <MetadataProvider id="SajatSPLocalMetadata"
       xsi:type="FilesystemMetadataProvider"
       metadataFile="%{idp.home}/metadata/sp-sp.intezmenyneve-metadata.xml"/>

<!-- ... további tartalom ... -->

</MetadataProvider>
```

EduID föderációs metaadat

EduID föderációs metaadat felvitele a `metadata-providers.xml` állományba az alábbiak szerint történik. Töltsük le föderáció metaadat aláíró tanúsítványát.

```
cd /opt/shibboleth-idp/credentials
wget https://metadata.eduid.hu/href-metadata-signer-2011.crt
```

A tanúsítvány SHA-1 és SHA-256 lenyomata a következőképpen ellenőrizhető le.

```
$ openssl x509 -in href-metadata-signer-2011.crt -noout -fingerprint
SHA1 Fingerprint=FE:AE:0B:E8:FB:59:ED:F7:CB:7F:69:DF:19:4F:8B:6D:C7:F6:96:66
$ openssl x509 -in href-metadata-signer-2011.crt -noout -fingerprint -sha256
SHA256
Fingerprint=B3:2C:16:EB:3B:77:39:42:46:E3:CD:D7:86:D6:0D:11:A9:FB:6E:1C:11:E2:FD:23:71:67:E4:3
3:B4:ED:9F:FA
```

Szerkesszük a beállítóállományt.

```
<MetadataProvider id="ShibbolethMetadata" xsi:type="ChainingMetadataProvider" ... >

<!-- ... további tartalom ... -->

    <!-- eduID.hu föderációs metaadat -->
    <MetadataProvider id="HTTPMetadata"
                  xsi:type="FileBackedHTTPMetadataProvider"
                  backingFile="%{idp.home}/metadata/localCopyFromEduID.xml"
                  metadataURL="https://metadata.eduid.hu/current/href.xml">
        <MetadataFilter xsi:type="SignatureValidation"
            requireSignedMetadata="true"
                        certificateFile="${idp.home}/credentials/href-metadata-signer-
2011.crt" />

<!-- ... további tartalom ... -->

</MetadataProvider>
```

Az eduID föderáció metaadat weblapja a https://metadata.eduid.hu/ helyen érhető el.

Tovább a föderációba

Amennyiben a telepítendő IdP-t szeretnénk a HREF-be integrálni, úgy ennél a pontnál küldjünk egy levelet az aai@niif.hu címre, amely nyomán, ha minden rendben van, az IdP regisztrálásra kerül a Resource Registry-ben.

Dokumentáció

- https://wiki.shibboleth.net/confluence/display/IDP30/MetadataConfiguration
- https://wiki.shibboleth.net/confluence/display/IDP30/HTTPMetadataProviders

Shib3IdpInstall

Az alábbiakban a Shibboleth 3 Identity Provider telepítése olvasható egyszerű beállítással.

A telepítés *Debian 8 (Jessie)* operációs rendszerre történik Jetty 9.2 alkalmazáskonténerbe.

Előkészületek

Telepítés előtt praktikus előkészíteni az Shibboleth 3 Identity Provider (IdP) környezetét. Az IdP központi szerepet tölt be, így elérhetősége szerencsés esetben ritkán változik.

- `entityID` (URI)
  - z entityID az a SAML azonosító, mely egyedi névvel látja el az IdP-t. Az első lépések egyike telepítéskor a megfelelő és gondos kiválasztása. Föderáción belül és világszinten egyedi kell legyen az ütközések elkerülése érdekében.
  - avasolt forma: **`https://idp.intezmenyneve.hu/idp/shibboleth`**.
  - gépnév rész legyen bejegyzett DNS név.
  - e tartalmazzon portszámot, lekérdezést, részazonosítót.
- `DNS`
  - Az entityID megtalálása után szükséges az IdP nevének DNS-be jegyzése.
  - avasolt forma: **`idp.intezmenyneve.hu`**.

> **Warning**
>
> Az entityID a szolgáltatás neve, nem a kiszolgáló számítógépé. Érdemes a neveket úgy megválasztani, hogy egy későbbi gépcsere esetén is független legyen a gépnév a szolgáltatás nevétől.

- `Tűzfal`
  - Szükséges nyitott portok a *TCP/443* és *TCP/8443*.
- `Operációs rendszer és Java futtató-környezet`
  - A jelen telepítési leírás Debian 8 (Jessie) rendszerhez készült.
  - Java telepítése
  - `apt-get install default-jdk`

Jetty 9.2 telepítés és előkészítés

Letöltés: http://download.eclipse.org/jetty/

Telepítés menete.

```
cd /opt
tar -xf jetty-distribution-9.2.<legutóbbi stabil verzió>.tar.gz
```

Jetty előkészítése Shibboleth 3 IdP számára.

```
cd /opt
mkdir jetty-shibboleth-idp
cd jetty-shibboleth-idp
mkdir etc modules lib logs resources webapps tmp
cd /opt/jetty-distribution-9.<legutóbbi stabil verzió>
cp etc/jetty-ssl.xml /opt/jetty-shibboleth-idp/etc/
cp etc/jetty-https.xml /opt/jetty-shibboleth-idp/etc/
cp etc/keystore /opt/jetty-shibboleth-idp/etc/
cp etc/keystore.pkf /opt/jetty-shibboleth-idp/etc/
cp modules/https.mod /opt/jetty-shibboleth-idp/modules/
cp modules/ssl.mod /opt/jetty-shibboleth-idp/modules/
```

Hozzuk létre a Jetty `start.ini` állományt az alábbi tartalommal az `/opt/jetty-shibboleth-idp/start.ini` helyen.

```
# Required Jetty modules
--module=server
--module=deploy
--module=annotations
--module=resources
--module=logging
--module=requestlog
--module=https
--module=ssl
--module=servlets
--module=jsp
--module=jstl
--module=ext
--module=plus

# Allows setting Java system properties (-Dname=value)
# and JVM flags (-X, -XX) in this file
```

```
# NOTE: spawns child Java process
--exec

-Didp.home=/opt/shibboleth-idp

-Djava.io.tmpdir=tmp

# Maximum amount of memory that Jetty may use, at least 512M is recommended
-Xmx512m

# Maximum amount of memory allowed for the JVM permanent generation
-XX:MaxPermSize=128m
```

Webapp XML állomány létrehozása az IDP-hez az `/opt/jetty-shibboleth/webapps/idp.xml` helyen az alábbi tartalommal.

```
<Configure class="org.eclipse.jetty.webapp.WebAppContext">
  <Set name="war"><SystemProperty name="idp.home"/>/war/idp.war</Set>
  <Set name="contextPath">/idp</Set>
  <Set name="extractWAR">false</Set>
  <Set name="copyWebDir">false</Set>
  <Set name="copyWebInf">true</Set>
</Configure>
```

Shibboleth 3 Identity Provider telepítés Jetty 9.2 konténerbe

Letöltés: http://shibboleth.net/downloads/identity-provider/latest/

```
cd /opt
wget http://shibboleth.net/downloads/identity-provider/latest/shibboleth-identity-provider-VERSION.tar.gz
tar -xf shibboleth-identity-provider-VERSION.tar.gz
```

Telepítés

```
root@idp:~# cd /opt/shibboleth-identity-provider-VERSION
root@idp:/opt/shibboleth-identity-provider-3.1.1# bin/install.sh
Source (Distribution) Directory: [/opt/shibboleth-identity-provider-3.1.1]

Installation Directory: [/opt/shibboleth-idp]

Hostname: [localhost.localdomain]
**idp.intezmenyneve.hu**
SAML EntityID: https://idp.intezmenyneve.hu/idp/shibboleth

Attribute Scope: [localdomain]
**intezmenyneve.hu**
TLS Private Key Password: ********
Re-enter password: ********
Cookie Encryption Key Password: ********
Re-enter password: ********
Warning: /opt/shibboleth-idp/bin does not exist.
Warning: /opt/shibboleth-idp/dist does not exist.
Warning: /opt/shibboleth-idp/doc does not exist.
Warning: /opt/shibboleth-idp/system does not exist.
Warning: /opt/shibboleth-idp/webapp does not exist.
Generating Signing Key, CN = idp.intezmenyneve.hu URI =
https://idp.intezmenyneve.hu/idp/shibboleth ...
...done
Creating Encryption Key, CN = idp.intezmenyneve.hu URI =
https://idp.intezmenyneve.hu/idp/shibboleth ...
...done
Creating TLS keystore, CN = idp.intezmenyneve.hu URI =
https://idp.intezmenyneve.hu/idp/shibboleth ...
...done
Creating cookie encryption key files...
...done
Rebuilding /opt/shibboleth-idp/war/idp.war ...
...done

BUILD SUCCESSFUL
Total time: 1 minute 21 seconds
root@idp:/opt/shibboleth-identity-provider-3.1.1#
```

Az IdP indítása

```
cd /opt/jetty-shibboleth-idp
java -jar /opt/jetty-distribution-9.2.<legutóbbi stabil verzió>/start.jar
```

Böngészőben a https://idp.intezmenyneve.hu:8443/idp URL-en rövid tájékoztatást kell kapnunk *No services are available at this location.* szöveggel.

Shibenv

Az alábbi test scriptek a Leuveni Egyetem Shibboleth oldaláról valók. (http://shib.kuleuven.be/download/sp/test_scripts/)

- PHP implementáció
- JSP (Java Server Pages) implementáció
- Lazy Session-t használó PHP implementáció (saját kiterjesztés)

ShibIdPAttrib

1. REDIRECT Attribútum feloldás

Shibboleth2 IdP

Telepítési leírások

- Shibboleth 2 IdP telepítése (Debian + Tomcat6)
- Shibboleth 2 IdP telepítése (RHEL5/CENTOS5 + Tomcat6)
- OpenSUSE-specifikus kiegészítések
- Shibboleth 2 IdP klaszterezése

Konfigurációs leírások

- Felhasználók azonosítása
- Attributumok feloldása
- Attributumok kiadása
- Perzisztens azonosító használata Shibboleth 2 IdP-ben
- Alkalmazásszerverhez delegált adatbázis kapcsolat kezelés (connection pooling) Shibboleth 2 IdP-ben
- Belépő oldal optimalizálása mobilra

Kiegészítő programok konfigurációs leírásai

- uApprove (ArpViewer) telepítése

Shibboleth fejlesztések(angolul) / Shibboleth IdP development (in English)

- Single Logout in Shibboleth
- Single Logout demo description
- X.509 authentication with LDAP
- Using Shibboleth SSO with webmail applications

Shib2SPSourceInstall

Függőségek

Fordítás

A Shibboleth fordításához néhány "külső" csomagot is le kell fordítani.

log4shib vs. log4cpp

> **Figyelem**
>
> Lásd: log4cpp kontra log4shib

XML-Security C

Ez a csomag igazából része általában a disztribúcióknak, de a Shibboleth 2-nek (és az OpenSAML-nak) >=1.4.0 verzió kell, **ami jelenleg még nincs is kiadva** (2008.04.24.). Hurrá :(

Örüljünk, a download könyvtárban meg lehet találni az 1.4.0 verziójú .tar.gz-t.

```
wget http://xml.apache.org/security/dist/c-library/xml-security-c-1.4.0.tar.gz
tar xzf xml-security-c-1.4.0.tar.gz
cd xml-security-c-1.4.0
./configure --prefix=/opt
make
sudo make install
```

xmltooling

Az xmltooling log4cpp-vel történő lefordításához szükség van erre a patch-re.

```
patch -p0 <../xmltooling_log4cpp5.patch
./configure --prefix=/opt --with-xmlsec=/opt
make
sudo make install
```

opensaml

```
./configure --prefix=/opt --with-xmlsec=/opt --with-xmltooling=/opt
make
sudo make install
```

Shibboleth SP

```
./configure --prefix=/opt --with-xmltooling=/opt --with-saml=/opt
make
sudo make install
```

Shib2PersistentId

Perzisztens azonosító használata Shibboleth2 IdP esetén

Ez az oldal a Shibboleth `storedId` plugin telepítését írja le MySQL adatbázis motorral.

Adatbázis driver telepítése

A MySQL JDBC driver a következő URL-ről érhető el: http://dev.mysql.com/downloads/#connector-j. Letöltés után a kitömörített drivercsomagból a .jar végű fájlt kell bemásolni a `/usr/share/tomcat6/lib` könyvtár alá.

Táblaszerkezet

```
CREATE TABLE `shibpid` (
  `localEntity` varchar(200) NOT NULL,
  `peerEntity` varchar(200) NOT NULL,
  `principalName` varchar(200) NOT NULL,
  `localId` varchar(200) NOT NULL,
  `persistentId` varchar(200) NOT NULL,
  `peerProvidedId` varchar(200) default NULL,
  `creationDate` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `deactivationDate` timestamp NULL default NULL,
  KEY `i_shibpid_pid` (`persistentId`),
  KEY `i_shibpid_pid_date` (`persistentId`,`deactivationDate`),
```

```
  KEY `i_shibpid_lid` (`localEntity`,`peerEntity`,`localId`),
  KEY `i_shibpid_lid_date` (`localEntity`,`peerEntity`,`localId`,`deactivationDate`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Perzisztens attribútum feloldása

Az attribute-resolver.xml konfigurációs fájlban vegyük fel a következő DataConnectort illetve PrincipalConnectort:

```
<resolver:DataConnector xsi:type="StoredId"
  xmlns="urn:mace:shibboleth:2.0:resolver:dc"
  id="persistentIdConnector"
  sourceAttributeID="uid"
  generatedAttributeID="persistentId"
  salt="valami-random-sztring">
  <resolver:Dependency ref="myLDAP" />
  <ApplicationManagedConnection jdbcDriver="com.mysql.jdbc.Driver"
     jdbcURL="jdbc:mysql://localhost/installfest"
     jdbcUserName="root"
     jdbcPassword="" />
</resolver:DataConnector>

<resolver:PrincipalConnector xsi:type="StoredId"
  xmlns="urn:mace:shibboleth:2.0:resolver:pc" id="saml2Persistent"
  nameIDFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
  storedIdDataConnectorRef="persistentIdConnector" />
```

Amennyiben alkalmazásszerver oldali kapcsolatkezelést szeretnénk használni, a következő leírás hasznos lehet.

Definiáljuk a persistentId attribútumot (a transientId ELŐTT! - különben a transientId -t választja ki az idp...):

```
<resolver:AttributeDefinition id="persistentId" xsi:type="Simple"
  xmlns="urn:mace:shibboleth:2.0:resolver:ad">
    <resolver:Dependency ref="persistentIdConnector" />
    <resolver:AttributeEncoder xsi:type="SAML2StringNameID"
      xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
```

```
      nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />
</resolver:AttributeDefinition>
```

Perzisztens attribútum kódolása a SAML válaszba

Az attribute_filter.xml -ben meg kell adni hogy a frissen létrehozott persistentId attribútumot kiadja az SP-nek:

```
<AttributeFilterPolicy id="releaseNameIDToAnyone">
    <PolicyRequirementRule xsi:type="basic:ANY" />

    <AttributeRule attributeID="transientId">
        <PermitValueRule xsi:type="basic:ANY" />
    </AttributeRule>
    <AttributeRule attributeID="persistentId">
        <PermitValueRule xsi:type="basic:ANY" />
    </AttributeRule>
</AttributeFilterPolicy>
```

Shib2IdpInstall

Előkészületek

entityID

Fontos, hogy az entityID **egyedi** és **állandó** legyen. Javasolt forma: `https://idp.intezmenyneve.hu/idp/shibboleth`.

Tűzfal

Be kell engedni a 443-as és a 8443-as portokat. Ha nagyon szigorúan vesszük, akkor a 8443-as portot elegendő csak a szóbajöhető SP-kről beengedni, de ezzel általában nem vagyunk tisztában, ezért célszerű a "nagyvilágból" beengedni. Biztonsági szempontból nem sok különbség van a 443-as és a 8443-as porton elérhető alkalmazások között.

JDK

Debian Lenny alatt a `sun-java6-jdk` csomagot kell feltelepíteni. Telepítés előtt érdemes az aptitude-ban kikapcsolni az opcionális függőségek telepítését.

```
aptitude install sun-java6-jdk
```

Állítsuk be, a `JAVA_HOME` környezeti változót!

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

> **Figyelem**
>
> Az `openjdk-6-jdk` csomag használata esetén ne felejtsük feltenni a `ca-certificates-java` csomagot, anélkül ugyanis hibát fogunk kapni az IdP indításakor!

Shibboleth security provider

> **Info**
>
> A Shibboleth Security Provider csak akkor szükséges, ha a Java alkalmazásszerver (Tomcat) önállóan fog kéréseket feldolgozni és nem kerül elé proxy (Apache)

Be kell másolni a `lib/shib-jce-1.0.jar` állományt a `$JAVA_HOME/jre/lib/ext` könyvtárba. Ha az `ext/` könyvtár nem létezik, akkor hozzuk létre.

```
cp lib/shib-jce-1.0.jar $JAVA_HOME/jre/lib/ext
```

Ezek után be kell állítani, hogy a JRE használni is tudja ezt a providert. Ehhez a `$JAVA_HOME/jre/lib/security/java.security` fájlban keressük meg az ún. "security provider"-eket, és írjuk hozzá a következő sort:

> security.provider.**7**
> =edu.internet2.middleware.shibboleth.DelegateToApplicationProvider

- **Megj.:** a "security.provider." után következő szám mindig a megelőzőnél legyen eggyel nagyobb!

Bouncy Castle JCE

> **Bizonytalan információ!**
>
> Az alábbi leírás az 5-ös JVM-hez készült, 6-os JVM esetén erre nem biztos hogy szükség van.

A JVM-mel jövő Java Cryptography Engine (JCE) nem támogatja az összes kriptográfiai algoritmust, amelyre az Identity Providernek szüksége lehet (pl. XML Digital Signature, XML Encryption). A Bouncy Castle JCE ezek mellett még olyan algoritmusokat is tartalmaz (általában hatékonyabb és szabványosabb formában), amelyek benne vannak a Java JCE-ben.

Ehhez először le kell tölteni a Bouncy Castle JCE-t. A JCE állományok a Provider oszlopban, a "Signed Jar Files" részben találhatók. (A nevük `bcprov-jdk-VERZIO.jar`.) Letöltés után a .jar fájlokat a `$JAVA_HOME/jre/lib/ext` könyvtárba kell tenni.

```
wget http://www.bouncycastle.org/download/bcprov-jdk15-141.jar
cp bcprov-jdk15-141.jar $JAVA_HOME/jre/lib/ext
```

Ezek után be kell állítani, hogy a JRE használni is tudja ezt a providert. Ehhez a `$JAVA_HOME/jre/lib/security/java.security` fájlban keressük meg az ún. "security provider"-eket, és írjuk hozzá a következő sort:

> security.provider.**8**=org.bouncycastle.jce.provider.BouncyCastleProvider

- **Megj.:** a "security.provider." után következő szám mindig a megelőzőnél legyen eggyel nagyobb!

Tomcat 6

A 2.1.3 és újabb IdP megköveteli a Tomcat6 használatát (Tomcat5.5 -tel bizonyos böngészők esetén nem működik rendesen). A Debian Lenny nem tartalmazza a Tomcat6-ot, ezért a testing ágból kell feltelepíteni.

A Tomcat6-ra való frissítésről további - vázlatos - információkkal ez az oldal szolgál.

Telepítés

Ha minden rendben meg, és a squeeze source beállításra került, akkor elegendő egy

```
aptitude install tomcat6
```

parancs kiadása. Ez felpakolja a tomcat különböző függőségeit is - az ajánlott függőségek (tomcat6-admin, -docs, stb.) feltelepítése nem szükséges.

Ne felejtsük el, hogy a Tomcat szerver "tomcat6" user nevében fog futni! Mivel a Shibboleth servletnek szüksége van arra, hogy hozzáférjen a filerendszerhez, a Java Security Manager-t ki kell kapcsolni a `/etc/default/tomcat6` fájlban:

```
TOMCAT6_SECURITY=no
```

Ahhoz, hogy a Tomcat számára üzembiztosan elegendő memóriát biztosítsunk, ugyanebbe a fájlba (`/etc/default/tomcat6`) adjuk meg:

```
JAVA_OPTS="-Xms256M -Xmx512M -XX:-DisableExplicitGC "
```

Beállítás

A `/etc/tomcat6/server.xml` fájlt kell szerkesztenünk

Ha a Tomcat Apache mögött fut

A 8009-es porton figyelő Connector elem konfigurációjához hozzá kell adni, hogy a `tomcatAuthentication` értéke "false" legyen, ezen kívül a hozzáférést korlátozhatjuk a localhost-ra is (hiszen a Connector-t csak a helyben futó Apache mod_proxy_ajp konnektora érheti el).

```
<Connector port="8009" address="127.0.0.1" tomcatAuthentication="false"
               enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

Ha a Tomcat önállóan, Apache nélkül fut

Ha a Tomcat Apache nélkül fut, akkor be kell állítani, hogy az SP-vel való kommunikációra fenntartott 8443-as porton egyből a Tomcat figyeljen.

```
<Connector port="8443"
           maxHttpHeaderSize="8192"
           maxSpareThreads="75"
           scheme="https"
           secure="true"
           clientAuth="want"
           SSLEnabled="true"
           sslProtocol="TLS"
           keystoreFile="IDP_HOME/credentials/idp.jks"
           keystorePass="PASSWORD"
           truststoreFile="IDP_HOME/credentials/idp.jks"
           truststorePass="PASSWORD"
           truststoreAlgorithm="DelegateToApplication"/>
```

Ahol az `IDP_HOME` az IdP alapkönyvtára, a `PASSWORD` pedig az IdP telepítésekor megadandó jelszó lesz.

> **Info**
>
> Amennyiben a Tomcat önállóan fut, szükség van a Shibboleth Security Provider telepítésére!

További információ angolul

Apache beállítás

Tanúsítványok beszerzése és bemásolása `/etc/ssl` vonatkozó alkönyvtárai alá.

Meg kell adni, hogy az Apache figylejen a 443-as és 8443-as portokon. Az alábbiak kerüljenek a `/etc/apache2/ports.conf` fájlba

```
Listen 443
Listen 8443
```

SSO URL (443-as port)

Be kell állítani a virtuális hosztot, amelyhez az IdP-t rendeltük. Először a 443-as portot konfiguráljuk.

```
<VirtualHost _default_:443>
  ServerName aai.example.org:443
  SSLEngine On
  SSLCertificateFile /etc/ssl/certs/aai.example.org.crt
  SSLCertificateKeyFile /etc/ssl/private/aai.example.org.key
  SSLCertificateChainFile /etc/ssl/certs/aai.example.org.crt
  ProxyRequests Off
  <Proxy ajp://localhost:8009>
    Allow from all
  </Proxy>
  ProxyPass /idp ajp://localhost:8009/idp retry=5
</VirtualHost>
```

Ezen a porton valamilyen széles körben ismert tanúsítványt kell használni, mivel a felhasználók böngészőjének ismerniük kell(ene) a kibocsátót.

AA ill. Artifact (8443-as port)

Ezen keresztül az SP és az IdP közvetlenül kommunikálnak egymással. Ide arra a tanúsítványra van szükség, amely a föderációs metadatában szerepel - az aláírója nem érdekes.

A csatorna felépítésekor az IdP és az SP is autentikálja magát. Az SP autentikációját az Apache végzi, ami nem végez kibocsátó-ellenőrzést (`optional_no_ca`). Ez utóbbit az IdP alkalmazás végzi el, ezért nagyon fontos, hogy a kliens tanúsítványát az Apache továbbadja az alkalmazásnak (`ExportCertData`).

> **Figyelem**
>
> Az Apache a tanúsítvány-ellenőrzésnél ellenőrzi a tanúsítvány típusát. Ezért az SP tanúsítványának vagy kliens tanúsítványnak kell lennie, vagy nem lehet benne típus információ.

```
<VirtualHost _default_:8443>
  ServerName aai.example.org:8443
  SSLEngine On
  SSLCipherSuite ALL:!ADH:!EXPORT56:!EXPORT40:RC4+RSA:!SSLv2:+HIGH:+MEDIUM:+LOW:+EXP
  SSLCertificateFile /etc/ssl/certs/aai-aa.example.org.crt
  SSLCertificateKeyFile /etc/ssl/private/aai-aa.example.org.key
  SSLVerifyDepth 10
  SSLVerifyClient optional_no_ca
  SSLOptions -StdEnvVars +ExportCertData
  ProxyRequests Off
  <Proxy ajp://localhost:8009>
    Allow from all
  </Proxy>
  ProxyPass /idp ajp://localhost:8009/idp retry=5
</VirtualHost>
```

A virtuális hoszt engedélyezése után be kell tölteni az `ssl` és `proxy_ajp` modulokat, majd újra kell indítani az apache-ot.

Shibboleth 2.x IdP servlet telepítés

Letöltés

A hivatalos IdP kiadás innen innen tölthető le

Alternatívaként az NIIF által patchelt, ezáltal SLO képes IdP kiadást ajánljuk, ami a NIIF AAI oldalról érhető el. A Single Logout-képes IdP-ről további információ itt.

Kicsomagolás

A `shibboleth-idp-2.x.x-bin.zip` fájl tartalma kicsomagolás után a `/usr/local/shibboleth-idp` könyvtár alákerül

```
cd /usr/local
jar -xf shibboleth-idp-2.x.x-bin.zip
```

Endorsed jar állományok

Sajnos - legalábbis a cikk írásakor - a "kincstári" Sun-os Tomcat (Java?) JAXP parser egy ismert memóriaszivárgást tartalmaz, ezért a disztribúcióban az `endorsed/` könyvtárban található .jar file-okat kézzel be kell másolni a Tomcat `endorsed/` könyvtárába.

- A Debian alatti tomcat6 csomag használatakor a `/usr/share/tomcat6/common/endorsed` könyvtárba kell tenni a jar file-okat (ezt a könyvtárt létre is kell hozni).
  mkdir /usr/share/tomcat6/endorsed cp endorsed/*.jar /usr/share/tomcat6/endorsed/

Installer

```
export JAVA_HOME=/usr/jdk
cd /usr/local/shibboleth-idp
chmod 755 install.sh
./install.sh
```

A telepítés során az alábbi kérdésekre kell választ adnunk:

> Is this a new installation? Answering yes will overwrite your current configurat ion. [yes|no]
> yes

Új telepítés, vagy sem.

> Where should the Shibboleth Identity Provider software be installed? /
> opt/shibboleth-idp-2.0.0
> /usr/local/shobboleth-idp

Itt található a letöltött és kicsomagolt shibboleth programcsomag

>

> What is the hostname of the Shibboleth Identity Provider server?
>
> idp.example.org
> idp.example.org

Shibboleth IdP alkalmazás URI alapú azonosítója.

> A keystore is about to be generated for you. Please enter a password that will be used to protect it.
> changeme

Feljegyzendő jelszó :)

Befejezés

Környezeti változó beállítása

```
IDP_HOME=/usr/local/shibboleth-idp
export IDP_HOME
```

Szimbolikus linkek megadása - az egyértelműség és konvenció kedvéért...

```
mv $IDP_HOME/conf /etc/`basename $IDP_HOME`
ln -s /etc/`basename $IDP_HOME` $IDP_HOME/conf
mv $IDP_HOME/logs /var/log/`basename $IDP_HOME`
ln -s /var/log/`basename $IDP_HOME` $IDP_HOME/logs
mkdir /var/lib/`basename $IDP_HOME`
mv $IDP_HOME/metadata /var/lib/`basename $IDP_HOME`/metadata
ln -s /var/lib/`basename $IDP_HOME`/metadata $IDP_HOME/metadata
```

Jogosultságok beállítása - hogy a `tomcat6` felhasználó hozzáférhessen az alábbi könyvtárakhoz

```
chown -R tomcat6 /var/log/`basename $IDP_HOME` /var/lib/`basename $IDP_HOME`
```

További, már telepített IdP-től függő tomcat beállítás

```
cd /var/lib/tomcat6/
mkdir -p conf/Catalina/localhost
```

Az így létrehozott könyvtárban készítsünk egy `idp.xml` nevű (a név legyen azonos a idp webalkalmazás nevével) fájlt az alábbi tartalommal:

```
<Context
        docBase="/usr/local/shibboleth-idp/war/idp.war"
        privileged="true"
        antiResourceLocking="false"
        antiJARLocking="false"
        unpackWAR="false" />
```

Naplófájlok rotálása

Az alapértelmezett logging.xml nem törli a régi állományokat, ezért ezek egy idő után megtöltik a diszket.

Erre a korrekt megoldás az (lenne), ha a Logback alrendszert utasítjuk, hogy az N (a példában 90) napnál régebbi fájlokat rotálja ki. Ehhez a `logging.xml`-ben adjuk meg a `maxHistory` paramétert az *összes* `rollingPolic`y-nál, valahogy így:

```
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
   <FileNamePattern>/usr/local/shibboleth-idp/logs/idp-access-%d{yyyy-MM-
dd}.log</FileNamePattern>
   <maxHistory>4</maxHistory>
 </rollingPolicy>
```

Sajnos azonban jelenleg a logback csak egy állományt töröl, a régi file-okat megtartja (pl. akkor is, ha több, mint egy napig nem futott az IdP. Amíg ez nincs megoldva, addig kerülő megoldás lehet cron-ból törölni a régi file-okat

```
sudo crontab -u tomcat6 -e


MAILTO=mail@example.com
#m h  dom mon dow   command
52 18 *   *   *     find /var/log/shibboleth-idp/ -mtime +90 -delete
```

Teszt

Ahhoz, hogy kiderítsük, működik-e (ill. fut-e :)) az IdP webalkalmazásunk, ahhoz böngészőben hívjuk meg az alábbi urlt: `https://idp.example.org/idp/profile/Status`, amennyiben az oldalon egy

`ok`-t látunk, akkor az alkalmazásunk fut, és elkezdhetjük beállítani az attribútumok feloldását és kiadását.

Ha nem működik a webalkalmazás, akkor az alábbi naplófájlokban kezdjünk el keresgélni:

- `/var/log/shibboleth/idp-error.log`
- `/var/log/shibboleth/idp-process.log`

A naplózás mélységét a `/etc/shibboleth/logging.xml` fájlban állíthatjuk be. Hibakereséshez érdemes a `<ErrorLog>` értékét `DEBUG`-ra állítani.

Shibboleth 2.0 IdP beállítás

Metadata beállítása

Metadata aláírás ellenőrzés beállítása

Az IdP-be beállított metaadat(ok) valódiságának ellenőrzéséhez szükséges egy ún. `TrustEngine` beállítása. Ezt a `relying-party.xml` -ben kell megtenni a `Security Configurations` részben:

```
<security:TrustEngine
   id="shibboleth.MetadataTrustEngine" xsi:type="security:StaticExplicitKeySignature">
 <security:Credential id="HREFSigner" xsi:type="security:X509Filesystem">
  <security:Certificate>/path/to/idp/credentials/href-metadata-signer-
2011.crt</security:Certificate>
 </security:Credential>
</security:TrustEngine>
```

A konfigurációban hivatkozott `href-metadata-signer-2011.crt` elérhető innen: https://metadata.eduid.hu/href-metadata-signer-2011.crt, SHA-1 lenyomata a következő: `FE:AE:0B:E8:FB:59:ED:F7:CB:7F:69:DF:19:4F:8B:6D:C7:F6:96:66`

HREF föderációs metadata beállítása az IdP-ben

A HREF metadata állomány elérhetősége:

- http://metadata.eduid.hu/current/href.xml

A Shibboleth IdP `relying-party.xml` konfigurációban a következőképpen lehet beállítani a HREF metaadatot (fontos hogy az előző pontban leírt `TrustEngine` is be legyen állítva):

```
<MetadataProvider id="HREF-Metadata"
     xsi:type="FileBackedHTTPMetadataProvider" xmlns="urn:mace:shibboleth:2.0:metadata"
     metadataURL="http://metadata.eduid.hu/current/href.xml"
     backingFile="/path/to/idp/metadata/href.xml"
     maxRefreshDelay="PT1H">
  <MetadataFilter xsi:type="SignatureValidation"
trustEngineRef="shibboleth.MetadataTrustEngine" />
</MetadataProvider>
```

Tovább a föderációba

Amennyiben a telepítendő IdP-t szeretnénk a HREF-be integrálni, úgy ennél a pontnál küldjünk egy levelet az aai@niif.hu címre, amely nyomán, ha minden rendben van, az IdP regisztrálásra kerül a Resource Registry-ben, s a válasz e-mail tartalmaz majd két hivatkozást, melyekről letölthetők az `attribute-filter.xml` és `attribute-resolver.xml` fájlok. Ezek már testreszabva tartalmazzák az IdP igényeit, az első fájlt elég csak bemásolni, a másodikban pedig - értelemszerűen - az egyes, a helyi erőforrásokra vonatkozó felhasználóineveket és jelszavakat kell kicserélni a megfelelőre.

További információk a Resource Registry-be történő felvételről

Attribútum filter automatikus frissítése
A Resource Registry automatikusan generálja minden egyes föderációban szereplő IdP számára a saját, testre szabott attribútum filterét, így célszerű úgy beállítani az IdP-t, hogy ezt a fájlt automatikusan töltse le. Ehhez hozzunk létre egy `confcache` nevű könyvtárat, adjunk rá írásjogot a `tomcat6` felhasználónak, majd szerkesszük a `conf/service.xml` fájlt. Az XML felső harmadában kerül megadásra az `AttributeFilterEngine`, melyet az alábbiak alapján kell átírni.

```
<Service id="shibboleth.AttributeFilterEngine"
       xsi:type="attribute-afp:ShibbolethAttributeFilteringEngine"
       configurationResourcePollingFrequency="3600000"
       configurationResourcePollingRetryAttempts="128">
      <ConfigurationResource url="https://rr.aai.niif.hu/gen_attribute-
filter.php/href/IDP_NEVE_AZ_RRBEN/attribute-filter.xml"
                             file="/path/to/shibboleth-idp/confcache/attribute-filter.xml"
xsi:type="resource:FileBackedHttpResource" />
</Service>
```

Több attribútum filter használata
Hasznos lehet, ha a föderációs szűrőkön kívül további irányokba kívánunk IdP-nkből attribútumokat kiadni. Elterjedt, hogy pl. különböző google szolgáltatásokhoz lehet az intézményen keresztül autentikálni, amely beállítási részleteket értelemszerűen a Resource Registry nem tartalmazza, így a letöltött friss, és a régi, helyi adatokat is tartalmazó fájlok egyesítése bosszantó plusz munka

lehet. Ezt elkerülendd dolgozhatunk több attribútum filterfájlból. Ehhez ismét a `conf/service.xml` fájlt kell szerkeszteni. Alább a fenti kódrészlet kiegészítése.

```
<Service id="shibboleth.AttributeFilterEngine"
       xsi:type="attribute-afp:ShibbolethAttributeFilteringEngine"
       configurationResourcePollingFrequency="3600000"
       configurationResourcePollingRetryAttempts="128">
      <ConfigurationResource url="https://rr.aai.niif.hu/gen_attribute-
filter.php/href/IDP_NEVE_AZ_RRBEN/attribute-filter.xml"
                             file="/path/to/shibboleth-idp/confcache/attribute-filter.xml"
xsi:type="resource:FileBackedHttpResource" />
      <ConfigurationResource file="/path/to/shibboleth-idp/conf/attribute-filter-local.xml"
xsi:type="resource:FilesystemResource" />
</Service>
```

Ezek a lépések természetesen kihagyhatók, ha nincs szándékunkban a föderáció tagjaivá válni, ekkor érdemes az alább részletezett attribútumokhoz kapcsolódó tudnivalókkal folytatni.

Statisztika küldés

Autentikáció beállítása

Attribútum feloldás beállítása

Attribútum kiadás beállítása

Shib2SPConfig

Az Shibboleth 2 SP-t a `shibboleth.xml` állományon keresztül konfigurálhatjuk. Ebben a leírásban feltételezzük, hogy az SP konfigurációja a `/etc/shibboleth` könyvtárban van.

Alapszerkezet

Mindenekelőtt megmutatjuk a `shibboleth.xml` fájl alapszerkezetét, majd alább az egyes szerkezeti elemeket részletesen is tárgyaljuk, majd a fejezet végén egy teljes, működő konfigurációt mutatunk be.

```
<SPConfig xmlns="urn:mace:shibboleth:sp:config:2.0"
    xmlns:conf="urn:mace:shibboleth:sp:config:2.0"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    logger="shibboleth/syslog.logger" clockSkew="180">

    <Extensions/>

    <OutOfProcess logger="shibboleth/shibd.logger"/>

    <InProcess logger="shibboleth/native.logger"/>

    <Listener/>

    <StorageService/>
    <SessionCache/>
    <ReplayCache/>
    <ArtifactMap/>

    <RequestMapper/>

    <ApplicationDefaults id="default" policyId="default"
        entityID="https://sp.example.org/shibboleth"
        homeURL="https://sp.example.org/index.html"/>

    <SecurityPolicies/>
```

```
</SPConfig>
```

Látható, hogy a szerkezet keretét egy `<SPConfig>` elem adja, ez fogja közre a különböző összetevők részletes konfigurációit. Az `<SPConfig>` opcionális attribútumai

- **`logger`** Annak a konfigurációs fájlnak a helyét adhatjuk meg, amelyben a loggolási tulajdonságok kerültek definiálásra. Alapértelmezés szerint ez a `shibboleth/shid.logger` fájl.
- **`clockSkew`** A legtöbb elosztott rendszerhez hasonlóan a Shibbolethnél is nagyon fontos, hogy szinkronban legyenek a rendszerben résztvevő elemek órái. Mivel komoly sebezhetőséget jelentene, ha a szerverek közti üzeneteken nem lenne megjelölve a feladás időpontja, ezért ezek az üzenetek időbélyeggel ellátottak, s minden rendszer elem csak egy bizonyos időnél nem régebbi üzenetekkel hajlandó foglalkozni. Ezt az értéket tudjuk itt megadni. Alapértelmezés szerint 3 perc, azaz 180 másodperc az értéke.

Összetevők

`<Extensions>`

`<OutOfProcess>`

`<InProcess>`

`<Listener>`

`<StorageService>`

`<SessionCache>`

`<ReplayCache>`

`<ArtifactMap>`

<RequestMapper>

A RequestMap megadja azokat a címeket (Host és Path), amelyeket a Shibboleth SP kezelni fog. Szerkezete:

```
<RequestMap applicationId="default">
    <Host name="www.example.org">
        <Path name="secure" authType="shibboleth" requireSession="true"/>
    </Host>
    <Host name="admin.example.org" applicationId="admin" authType="shibboleth"
requireSession="true">
        <AccessControl>
            <Rule require="affiliation">faculty@osu.edu student@osu.edu</Rule>
        </AccessControl>
    </Host>
</RequestMap>
```

A RequestMap több Host elemet is tartalmazhat, a Host elem 0 vagy több Path elemet tartalmazhat.

!!! danger "Figyelem"

```
Ha 1-nél nagyobb mélységű könyvtárat (pl. a `/shibtest/shibreq` nevűt) szeretnénk védeni,
akkor **nem** adhatjuk meg a *name* paraméterben a "shibtest/shibreq" értéket, hanem egymásba
ágyazott Path elemeket kell használni. A *name* paraméter nem tartalmazhat '/' karaktert.
```

Az egyes elemeknél paraméterekkel szabályozhatjuk, hogy az SP milyen módon kezelje a hostot vagy az útvonalat. A paraméterek felüldefiniálhatók. A legfontosabb paraméterek az alábbiak (ezek ugyanúgy használhatók Host-nál mint Path-nál):

- **`requireSession`**: ha értéke "true", akkor az SP csak akkor továbbítja a HTTP request-et az alkalmazás ill. a webszerver felé, ha sikerült létrehozni egy autentikált session-t. Ha "false", akkor az alkalmazás felelős azért, hogy létrehozza a Shibboleth session-t (ún. lazy session) Alapértelmezés: "false"
- **`applicationId`**: lehetőség van arra, hogy bizonyos helyekre érkező kérésekre az SP más és más módon próbáljon meg session-t létrehozni, ezt ún. Shibboleth Application-ben konfigurálhatjuk. Ha nem adunk meg értéket, akkor a "default" application-nél megadott értékek vonatkoznak majd a session-re.

<ApplicationDefaults>

ShibSPInstallDebian

Ez egy elavult lap, használd ezt helyette!

A Debian 4.0 (Etch) már tartalmazza a Shibboleth SP-t (és függőségeit), ezért csak a megfelelő csomagokat kell telepítenünk.

- **Ez igaz, csakhogy a debianos liblog4cpp4 nem thread-safe, ezért a shibd nagyobb terhelésnél elhasal! A probléma javítása folyamatban van...**

```
root@hal:~# <b>apt-get install libapache2-mod-shib opensaml-schemas</b>
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Az alábbi extra csomagok kerülnek telepítésre:
libicu36 liblog4cpp4 libsaml5 libshib-target5 libshib6 libxalan110
libxerces27 libxml-security-c12
Javasolt csomagok:
xalan
Az alábbi ÚJ csomagok lesznek telepítve:
libapache2-mod-shib libicu36 liblog4cpp4 libsaml5 libshib-target5 libshib6
libxalan110 libxerces27 libxml-security-c12 opensaml-schemas
0 frissített, 10 újonnan telepített, 0 eltávolítandó és 18 nem frissített.
Letöltés az archívumokból: 12,7MB
Kicsomagolás után 39,6MB lemezterületet használok fel
Folytatni akarod [Y/n]? <b>y</b>
```

A telepítés után az Apache webszervert újra kell indítani.

Shibboleth Service Provider SP és Docker

Az alábbi lapon összefoglaljuk a legfontosabb lépéseket, melyek általános esetben elegendőek ahhoz, hogy működő Shibboleth SP-t állítsunk üzembe, Docker konténerben. Fontos, hogy rengeteg olyan igény lehet, amely további speciális beállítások meglétét teszik szükségessé, ezeket ezen a lapon nem részletezzük, ilyen irányú tájékozódáshoz legbiztosabb források:

- http://wiki.shibboleth.net/
- https://docs.docker.com/

Apache 2.4

Az alábbi példákban az SP és az alkalmazás konténer SSL termination proxy mögött helyezkedik el. Természetesen a VirtualHost átalakítható úgy, hogy SSL-t is ki tudjon szolgálni, ha erre van igény.

Proxy

Ebben az esetben, a Shibboleth SP-vel védeni kívánt alkalmazást proxy-zuk egy másik futó konténerből.

```
<VirtualHost *:80>
    ServerName https://domain:443
    ServerAdmin admin@domain.com
    UseCanonicalName On
    ErrorLog /dev/stderr
    CustomLog /dev/stdout combined
    ProxyVia On
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass /Shibboleth.sso !
    ProxyPass / http://cel_kontener:8080/ retry=0 timeout=30
    ProxyPassReverse / http://cel_kontener:8080/
```

```
    <Location "/socket.io">
        RewriteEngine On
        RewriteCond %{QUERY_STRING} transport=websocket [NC]
        RewriteRule /(.*) ws://cel_kontener:8080/socket.io/$1 [P,L]
        ProxyPass http://cel_kontener:8080/socket.io retry=0 timeout=30
        ProxyPassReverse http://cel_kontener:8080/socket.io
    </Location>
    <Proxy "*">
        AuthType shibboleth
        ShibRequestSetting requireSession 1
        Require valid-user
    </Proxy>
</VirtualHost>
```

Lazy session

Az előző példához hasonlóan, a Shibboleth SP-vel védeni kívánt alkalmazást proxy-zuk egy másik futó konténerből, de csak a https://domain.com/secure útvonalat védjük.

```
<VirtualHost *:80>
    ServerName https://domain:443
    ServerAdmin admin@domain.com
    UseCanonicalName On
    ErrorLog /dev/stderr
    CustomLog /dev/stdout combined
    ProxyVia On
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass /Shibboleth.sso !
    ProxyPass / http://cel_kontener:8080/ retry=0 timeout=30
    ProxyPassReverse / http://cel_kontener:8080/
    <Location "/secure">
        AuthType Shibboleth
        ShibRequestSetting requireSession true
        ShibUseHeaders On
        Require shibboleth
        ProxyPass http://cel_kontener:8080/secure retry=0 timeout=30
        ProxyPassReverse http://cel_kontener:8080/secure
```

```
    </Location>
    <Proxy "*">
        AuthType shibboleth
        ShibRequestSetting requireSession false
        Require shibboleth
    </Proxy>
</VirtualHost>
```

Shibboleth

Load balance

Terheléselosztó mögött a **shibboleth2.xml**-ben, a `<Session>` beállításnál érdemes a `consistentAddress="false"` értéket beállítani, ha tudjuk, hogy változó (LAN!) címekről érkeznek a felhasználók.

Shibboleth

Architektúra

Profilok

- Shibboleth POST Profile
- Shibboleth Artifact Profile
- Lazy Session
- Attribute Push

Jelentés

Shib2IdpConnectionPool

Mi is az a connection pooling?

A Java webalkalmazások általában többszálú, hosszú életciklusú környezetben futnak, az adatbázis kapcsolatok azonban természetükből fakadóan nem szálbiztosak (egy kapcsolaton egyszerre csak szál dolgozhat). Ezért szükség van arra, hogy a feldolgozó szálakhoz adatbázis kapcsolatokat rendeljünk. Megjegyzendő, hogy ez nem újdonság, ugyanis PHP esetén például minden egyes futáskor új kapcsolat nyílik.

A kapcsolatkezelésre tehát alapvetően három módszer ismert:

- egy adatbáziskapcsolat, szinkronizációval. Ebben az esetben az alkalmazásunk tulajdonképpen egyszálúvá válik.
- feldolgozó szálanként külön adatbáziskapcsolat, így nem kell a szinkronizációval foglalkozni. Így azonban a kapcsolatok jelentős része kihasználatlan, szélsőséges esetben az adatbázis által engedélyezett kapcsolatok száma telítődhet is. Ez a megoldást tehát jelentős overheadet okoz.
- az adatbázis kapcsolatok dinamikus kiosztása a feldolgozó szálak között.

A fenti három megoldás közül az első kettő webes környezetben teljesítmény okokból erősen ellenjavallt, csak a harmadik módszer a járható út: az adatbázis kapcsolatokat tehát érdemes "készletezni" (connection pooling), és a kapcsolatot kérő szálakhoz futás közben, igény szerint hozzárendelni őket.

Ezen paradigma hatékony működéséhez rendkívül fontos, hogy az alkalmazás csak annyi ideig használja a kapcsolatot, ameddig szükséges, majd azonnal jelezze a pool felé, hogy a kapcsolat szabad (ezt tulajdonképpen a kapcsolat API-szintű lezárásával jelzi, de ilyenkor a fizikai kapcsolat természetesen nem bomlik fel, az másik szál számára ismét kiajánlhatóvá válik).

Az elterjedtebb connection pool implementációk rengeteg segítséget képesek nyújtani:

- minimális és maximális kapcsolatszám meghatározása
- inaktivitás esetén lezárás
- adatbázis oldali lezárás megakadályozása folyamatos "pingeléssel"
- halott kapcsolatok transzparens eltávolítása

A Java alkalmazásszervereknek egy szabványos módszert kell biztosítaniuk az adatbázis kapcsolatok elérésére. Az alkalmazások egy ún. JNDI erőforráson keresztül képesek elérni a kapcsolatok menedzseléséért felelős osztályt (ami általában egy ún. DataSource interfészt implementál). Fontos megemlíteni, hogy ilyenkor az adatbáziskapcsolat felépítését az alkalmazásszerver végzi, így az elérés paramétereit (url, felhasználónév, jelszó) ott kell adminisztrálni, és nem az alkalmazás saját konfigurációjában. Ez lehetőséget ad az adminisztrátor számára, hogy az egyes alkalmazásspecifikus beállítások megtanulása nélkül képes legyen egyik adatbázisról a másikra migrálni.

"Hivatalos", Sun-féle leírás

Connection pool használata Tomcat6 alatt

A Tomcat jelenleg a dbcp nevű pool implementációt használja, ami elég régi és a teljesítménye sem a legjobb, de legalább működik.

Az alábbi példakód egy MySQL kapcsolatot állít be (`/etc/tomcat6/server.xml`), ami kapcsolat-ellenőrzést is végez, mielőtt az alkalmazásnak válaszolna:

```
  <GlobalNamingResources>

    <!-- Create connection pool for idptest.
         Connections will be validated before handed over to the application,
         and every 10 minutes. -->
    <Resource name="jdbc/mymysql" auth="Container"
              type="javax.sql.DataSource"
              driverClassName="com.mysql.jdbc.Driver"
              maxActive="10" maxIdle="2" maxWait="10000"
              username="username" password="password"
              url="jdbc:mysql://localhost:3306/database"
              testWhileIdle="true" timeBetweenEvictionRunsMillis="6000000"
              testOnBorrow="true" validationQuery="SELECT 1" />
  </GlobalNamingResources>
```

A fenti globális JDNI erőforrást egyes alkalmazásokhoz kell rendelni a Context leíróban (pl `/etc/tomcat6/Catalina/localhost/idp.xml`):

```
<Context docBase="...." ... >
   <ResourceLink name="jdbc/mymysql"
                 global="jdbc/mymysql"
                 type="javax.sql.DataSource" />
</Context>
```

A fenti konfiguráció elkészülte után az alkalmazás a `java:comp/env/jdbc/mymysql` JNDI néven éri el az adatbázis kapcsolatot, valahogy így (hibakezelés nélkül, természetesen):

```
// initialize jndi datasource
Context ctx = new InitialContext();
DataSource dataSource = (DataSource) ctx.lookup("java:comp/env/jdbc/mymysql");
// acquire a new connection
Connection conn = dataSource.getConnection();
try {
  // do something
} finally {
  //don't forget to close the connection in the finally block!
  conn.close();
}
```

Egy fontos megjegyzés

Az alkalmazásszerver és az alkalmazás általában két külön osztálybetöltőt (classloader) használ, ezért ebben az esetben nem az alkalmazás mellé kell csomagolni a megfelelő adatbázis drivert, hanem az alkalmazásszerver által látható helyre. Debian Lenny alatt ez például a következőképpen valósítható meg:

```
sudo aptitude install libmysql-java
sudo ln -s /usr/share/java/mysql.jar /usr/share/tomcat6/lib/
```

IdP konfigurálása

Tegyük fel, hogy alkalmazásszerverünk képes a connection poolingra, és a megfelelő DataSource objektumot a JNDI térben rendelkezésre bocsátja `jdbc/idp` néven.

Attribútumok feloldása

Az IdP leggyakrabban attribútum feloldáshoz (pl. perzisztens azonosítókhoz) használ relációs adatbázist, ezért példaként álljon itt egy DataConnector konfiguráció:

```
<resolver:DataConnector xsi:type="StoredId" ...>
  <resolver:Dependency ref="myLDAP" />
  <ContainerManagedConnection resourceName="java:comp/env/jdbc/idp" />
</resolver:DataConnector>
```

Gyakorlatilag az `ApplicationManagedConnection` mondásokat kell `ContainerManagedConnection`-ra cserélni.

JDBC login modul

Bővebben lásd: Shib2IdpAuth#SQL (JDBC) alapon

uApprove

Az alábbi leírás a uApprove legalább 2.3-as verzióihoz íródtak, korábbiak használata nem javasolt.

Ahhoz, hogy a tomcat által kezelt adatbáziskapcsolat használatára rávegyük a uApprove-ot, az alábbiakat kell tennünk.

```
vim conf/uApprove.properties
```

Az adatbázisbeállításoknál kommentezzük ki a default beállításokat és írjuk be a használni kívánt JNDI resource nevét:

```
#--------------------------------------------------------------------#
# Database configuration                                             #
#--------------------------------------------------------------------#

database.resourceName       = java:comp/env/jdbc/mymysql
#database.driver             = com.mysql.jdbc.Driver
#database.url                = jdbc:mysql://localhost:3306/uApprove
#database.username           = uapprove
#database.password           = password

vim conf/uApprove.xml
```

Tegyük inaktívvá az alapértelmezett uApprove.dataSource bean-t, és helyére tegyük az alábbi definíciót

```
<bean id="uApprove.dataSource" class="org.springframework.jndi.JndiObjectFactoryBean" depends-on="shibboleth.LogbackLogging"
    p:jndiName="${database.resourceName}" p:lookupOnStartup="true"
    p:cache="true" p:proxyInterface="javax.sql.DataSource" />
```

Utolsó lépésként másoljuk be a `shibboleth-idp/lib` és a `shibboleth-idp/war/WEB-INF/lib` könyvtárba az alábbi két jart is. (Alapértelmezés szerint az utóbbi útvonal becsomagolva található a shibboleth-idp/war/idp.war fájlban, ezesetben kicsomagolás --> fájlok bemásolása --> visszacsomagolás a követendő út):

- aopalliance-1.0.jar
- spring-aop-3.0.6.RELEASE.jar

ShibIdPX509LdapAuthentication

Shibboleth 2.x IdP X.509/LDAP autentikációs modul

Ezen az oldalon az NIIF által fejlesztett X.509 klienstanúsítvány alapú Shibboleth autentikációs modul leírása szerepel.

In English

Motivations

- The use of hardware tokens as authentication source. However, X.509 certificate authentication is not generally considered secure by nature, hardware tokens are designed to be safer than passwords. Local policy can decide whether they accept software tokens or not.
- Give the choice to our SPs. Some SPs can decide if they wanted to force the X.509 authentication (or force password authentication).

PKIful versus PKIless

- If one has built their full-fledged PKI infrastructure, one could use it for client certificate authentication.
- But it is hard to do PKI right, CRLs and/or OCSP are crucial in PKI.
- If only authentication is needed, storing the (self-signed) certificate is enough.

Shibboleth X.509 authentication

- With PKI, you would use simple RemoteUser authentication with container support.
- Without PKI, the container can not authenticate the user, it can only check if the user has the corresponding private key.
- You will also need some custom workflow to validate the presented client certificates. Eg. checking them against the directory attribute 'userCertificate'. This is step is a must to have control over certificate revocation.

X.509 + LDAP certificate authentication (implementation details)

- The web container does client certificate checking, but does not validate. Instead, it handles the certificate to the Shibboleth authentication module which will validate it.
- Shibboleth is configured with RemoteUser login handler pointing to our X.509 authentication servlet.
- The certificate must contain some identification data (eg the X.500 'UID' RDN). Our authentication servlet takes the presented certificate and compares it to the stored certificate(s) for the user. If a matching certificate is found in the directory, then the user is authenticated.
- The certificate authentication is implemented as a standard JAAS module, and can be reused elsewhere.

Combining X.509 and username/password authentication

- When SP does not specifically request authentication methods, the user should have the choice between supported authentication modes. Otherwise, the IdP must conform with the authentication context class the SP sent. The IdP must refuse to authenticate the user with authentication methods unacceptable to the SP. There is a support ticket named SIDP-258 about this flaw in Shibboleth IdP.
- We want to support two authentication methods: username/password (PasswordProtectedTransport) and X.509 authentication.
- Unfortunately this is not enough, we need a default authentication method which offers the choice of these two methods to our users. This can be done by placing a link to the X.509 authentication servlet in login.jsp. However when the SP requests PasswordProtectedTransport, this link must not be visible, so we decided to configure a new UserPassword login handler which maps to the unspecified authentication class and uses this tweaked login.jsp.

- We also want to send the actual authentication method to the SP (instead of saying 'unspecified'), so both login handlers must set their corresponding authentication class in the Shibboleth request. As the internal UsernamePassword login servlet does not do this, we subclassed it.
- Playing with Shibboleth login handlers and authentication contexts revealed that Shibboleth IdP can not properly support default authentication methods, and our hybrid handler with its 'unspecified' authentication method is invoked on every authentication request (because both actual methods it uses override this unspecified method in the request and IdP can not decide whether the unspecified class is requested by the SP or it is simply the default method configured in relying-party.xml). Fixing SIDP-265 with our proposed patch corrected this behavior.

Követelmények

Az X.509/LDAP autentikációs modul a következő követelmények alapján került kifejlesztésre:

- a felhasználók saját maguk által aláírt tanúsítványokat is használhassanak autentikációra
- ne kelljen PKI infrastruktúrát üzemeltetni a klienstanúsítványok használatához
- a tanúsítványok központilag menedzseltek, egyszerűen visszavonhatók legyenek

Ezen követelmények kielégíthetők a címtárban tárolt klienstanúsítványokkal, ugyanis a címtárba csak egy felettes szerv képes beírni a tanúsítványt, ott minden bejelentkezéskor ellenőrzésre is kerül, ezért könnyen visszavonható.

> **Info**
>
> A felhasználó tanúsítvány alapú azonosításához (identification) szükséges, hogy a tanúsítvány tartalmazza a felhasználónevet, mégpedig az `UID` (subject) mezőben.

Telepítés

Az autentikációs modul letölthető a http://software.niif.hu oldalról. A Shibboleth2 IdP autentikációs motorjának konfigurációját részetesen a Shibboleth2 User Authentication wikioldal írja le.

Apache beállítása

Amennyiben az alapértelmezett szervlet elérési utat választjuk (`/Authn/X509`), a következő opciókat kell megadni az Apache webszerver konfigurációjában:

```
<Location /idp/Authn/X509>
   SSLVerifyClient optional_no_ca  #nincs CA ellenorzes
   SSLOptions +ExportCertData      #tanusitvany exportalasa
</Location>
```

IdP webalkalmazás beállítása

A letöltött modulban található `shibboleth-x509auth-verzio.jar` java osztálykönyvtárat be kell másolni a Shibboleth webalkalmazás `WEB-INF/lib` könyvtárába, valamint a `WEB-INF/web.xml` fájlban meg kell adni az autentikációs szervlet paramétereit:

```
<servlet>
  <servlet-name>X509LdapAuthHandler</servlet-name>
  <servlet-class>hu.niif.middleware.shibboleth.auth.X509LdapLoginServlet</servlet-class>
  <init-param>
    <param-name>jaasConfigName</param-name>
    <param-value>X509LdapAuth</param-value>
  </init-param>
  <load-on-startup>4</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>X509LdapAuthHandler</servlet-name>
  <url-pattern>/Authn/X509</url-pattern>
</servlet-mapping>
```

> **Info**
>
> Az IdP webalkalmazás az `${SHIB_HOME}/war/idp.war` fájlban található, ebben kell elvégezni a módosításokat, majd újratelepíteni az alkalmazást a webkonténerbe.

IdP konfiguráció

Az IdP konfigurációjában két dolgot kell módosítani: a JAAS autentikációs modul `login.config` konfigurációját, valamint a `handler.xml`-ben az autentikációs módokat.

Az X.509/LDAP JAAS modul beállításához a `${SHIB_HOME}/conf/login.config` fájl tartalmához a következő sorokat adjuk hozzá (az LDAP elérési paramétereket értelem szerint kitöltve; az értékek általában a `ShibUserPassAuth` JAAS konfigurációból átmásolhatóak):

```
X509LdapAuth {
  hu.niif.middleware.jaas.X509LdapLoginModule required
     host=""
     port=""
     base=""
     ssl=""
     userField=""
     serviceUser=""
     serviceCredential="";
};
```

> **Info**
>
> Figyelni kell arra, hogy az itt megadott `serviceUser` olvasási joggal rendelkezzen a `userCertificate` LDAP attribútumra.

A JAAS modul beállítása után a `${SHIB_HOME}/conf/handler.xml` fájlban meg kell adnunk az új autentikációs modulunkat, a következőképpen:

```
<LoginHandler xsi:type="RemoteUser" protectedServletPath="/Authn/X509" >
   <AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:X509</AuthenticationMethod>
</LoginHandler>
```

Ha továbbra is alapértelmezetten felhasználónév/jelszó autentikációt szeretnénk használni, akkor a Shibboleth IdP-ben be kell állítani az alapértelmezett hitelesítési módot, a `${SHIB_HOME}/conf/relying_party.xml` fájlban:

```
...
<DefaultRelyingParty provider="..."
  defaultSigningCredentialRef="..."

defaultAuthenticationMethod="urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
">
...
```

Shibboleth SP beállítása

Az egyes SP-k eldönthetik hogy ők kérik-e az X.509 autentikációt, ezt az `authnContextClassRef` SP opcióval lehet jelezni a Shibboleth SP felé.

Ez a kérés azonban nem teljesen megbízható, ezért érdemes az IdP oldalon konfigurálni hogy egy adott SP kérése alapján az X.509 autentikációs módot használjuk (a `${SHIB_HOME}/conf/relying-party.xml` fájlban):

```
...
<RelyingParty id="x509-protected-sp-entityid"
  provider="our-idp-entityid"
  defaultAuthenticationMethod="urn:oasis:names:tc:SAML:2.0:ac:classes:X509" />
...
```

Integráció a felhasználónév / jelszó bejelentkezéssel

A fent leírt telepítési módszer nem biztosítja azt a lehetőséget, hogy egy felhasználó eldönthesse hogy ő jelszóval vagy klienstanúsítvánnyal autentikál. Amennyiben szeretnénk ezt a választási lehetőséget megadni abban az esetben, amikor az SP nem jelez általa preferált autentikációs módot, az alábbi plusz konfiguráció segítségével ezt megtehetjük.

> **Info**
>
> A klienstanúsítvány segítségével történő autentikáció **nem feltétlenül biztonságosabb** mint a jelszavas bejelentkezés, ezért jól fontoljuk meg, hogy minden felhasználónak felkínáljuk-e ezt a lehetőséget.

Shibboleth IdP webalkalmazás módosítása

Az X.509/LDAP autentikációs modul tartalmaz egy olyan szervletet, ami képes a felhasználónév/jelszó és az X.509 autentikáció együtt történő futtatására. Első lépésként ezt a szervletet kell beállítani a `WEB-INF/web.xml` webalkalmazás konfigurációban:

```
<servlet>
  <servlet-name>UsernamePasswordX509LoginServlet</servlet-name>
  <servlet-class>hu.niif.middleware.shibboleth.auth.UsernamePasswordX509LoginServlet</servlet-class>
  <init-param>
    <param-name>loginPage</param-name>
    <param-value>login_.jsp</param-value>
```

```
  </init-param>
  <load-on-startup>4</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>UsernamePasswordX509LoginServlet</servlet-name>
  <url-pattern>/Authn/UserPasswordX509</url-pattern>
</servlet-mapping>
```

Ez a konfiguráció hivatkozik a `login_.jsp` fájlra, ez egy módosított Shibboleth bejelentkeztető form, amiben két plusz gomb kapott helyet. Ezekkel a gombokkal a felhasználó kérheti, hogy erre az egy autentikációra szeretne klienstanúsítványt használni, vagy a munkamenetben mindig. Utóbbi esetben a bejelentkezést lekezelő szervlet létrehoz egy cookie-t a felhasználó gépén, ami ezt a preferenciát megőrzi a böngésző bezárásáig.

> **Info**
>
> Amennyiben a felhasználó egyszer bejelölte a tanúsítványos autentikációt a teljes munkamenetre, azt nem tudja kikapcsolni, csak a böngésző újraindításával.

Shibboleth IdP konfiguráció

Az IdP konfigurációjában meg kell adni ezt a hibrid autentikációs módot, mégpedig a következőképpen (`${SHIB_HOME}/conf/handler.xml`):

```
<LoginHandler xsi:type="UsernamePassword"
  authenticationDuration="240"
  jaasConfigurationLocation="file://PATH/TO/IDP/conf/login.config"
  authenticationServletURL="/Authn/UserPasswordX509">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</AuthenticationMethod
>
</LoginHandler>
```

Ez a konfigurációs részlet azt közli az IdP-vel, hogy az autentikációs modul nem specifikált autentikációs mód esetén működik. Ezt a módot alapértelmezetté tehetjük a `${SHIB_HOME}/conf/relying_party.xml` fájlban:

```
...
<DefaultRelyingParty provider="..."
  defaultSigningCredentialRef="..."
```

```
  defaultAuthenticationMethod="urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified">
...
```

Shibboleth SP

Telepítési leírások

- Shibboleth SP telepítés

Konfigurációs leírások

- SP alkalmazás konfigurációja
- Attribútumok használata
- Naplózási beállítások
- Alkalmazás levédése Shibboleth-tel
- Lazy Session használata

- Új SP hozzáadása a föderációhoz

- Tesztelés

Shib2IdpCluster

Shibboleth 2.1 IdP klaszterezése

Klaszter terminológia

Node

- egy, a szolgáltatást futtató csomópont

Klaszter

- kívülről nem megkülönböztethető nodeok összessége

Szerver

- fizikai (vagy virtuális) gép, amely a nodeokat futtatja (egy gépen lehet több node is)

Failover

- amennyiben egy csomópont kiesik, egy másik csomópont automatikusan és transzparensen átveszi a munkáját

High availability

- amennyiben egy csomópont kiesik, nem veszhet el adat, a kliensek nem veszik észre a kiesést

Load balancing

- a terhelés elosztása az egyes csomópontok között

Terracotta

A Shibboleth2 IdP a Terracotta szoftvert támogatja a klaszter építéséhez. A Terracotta képes arra, hogy a különböző nodeokon futó Shibboleth IdP-k között a session és egyéb információkat (például artifact map, authnrequest replay map, stb.) szinkronban tartsa.

A Terracotta kliens-szerver architektúrában működik. A kliensek a JVM-ben futó instrumentált osztályokból, osztálybetöltőből és zármenedzserből állnak, a szerverek pedig biztosítják a klaszterezett adatok perzisztens tárolását és a csomópont-független elérést. Amennyiben egy JVM-ben szükség van egy távoli JVM-ben létrehozott klaszterezett objektumra, akkor ezt az első elérésnél a Terracotta kliens elkéri a távoli szervertől. Emiatt teljesítmény okokból érdemes az azonos felhasználóhoz tartozó kéréseket mindig ugyanahhoz a csomóponthoz küldeni. A HTTP-Artifact profil használata esetén ez nem garantálható, ezért ajánlott a HTTP-Post profil használata.

Shibboleth IdP és Terracotta

A Shibboleth IdP-ben a következő adatok klaszterezését kell megvalósítani: artifact, session, replay, transientId, loginContext. Ezeket az adatokat a Shibboleth StorageService tárolja.

A Terracotta telepítéséhez és beállításához ez a wikioldal nyújt segítséget.

1. Terracotta letöltése, kicsomagolása
2. tűzfalbeállítások (TCP/9510 kliens->szerver és TCP/9530 szerver->szerver portok engedélyezése)
3. minden csomóponton azonos JVM verziót használjunk a Terracotta szerverben és kliensben is
4. tim-vector integrációs modul telepítése
5. Shibboleth IdP-hez Terracotta konfiguráció szerkesztése [[Shib2IdpTerracottaConfiguration]] alapján
   - csomópontok definiálása (szervernév, hosztnév, logok helye)
6. terracotta szerver futtatása
7. boot jar készítése (Terracotta kliens)
8. boot jar használata a webkonténer JVM-nél
9. **FONTOS: JVM frissítés után újra kell generálni a boot jart!**

JVM beállítások:

```
JAVA_OPTS="-Dtc.install-root=$TC_INSTALL_DIR \
           -Dtc.config=$SHIB_IDP_HOME/conf/tc-config.xml \
           -Xbootclasspath/p:$TC_INSTALL_DIR/lib/dso-boot/dso-boot-hotspot_$jvm_spec_ver.jar"
```

2.1.2 -es IdP verzióhoz a következő konfigurációs rész is szükséges az instrumented-classes szekcióba:

```
  <include>
    <class-expression>org.opensaml.xml.util.LazyList</class-expression>
    <honor-transient>true</honor-transient>
  </include>
```

A Terracotta log- és adatfájljainak /var alatti tárolását a következőképpen végezhetjük el:

Könyvtárak létrehozása megfelelő jogosultságokkal

```
mkdir /var/{lib,log}/terracotta/{client,server}
chown tomcat55.nogroup /var/{lib,log}/terracotta/client
chown nobody.nogroup /var/{lib,log}/terracotta/server
```

Terracotta tc-config.xml -ben a data,stats,logs opciók átírása értelem szerint.

Magas rendelkezésre állás beálítása

A következő konfigurációs részt a tc-config.xml elején kell elhelyezni. Ez engedélyezi a kliens-szerver és szerver-szerver újrakapcsolódást, ezzel kivédve az apró hálózati kimaradások okozta problémákat. Sajnos mindkét beállítás megnöveli a failover idejét.

```
  <tc-properties>
   <!-- server-to-server reconnect -->
   <property name="l2.nha.tcgroupcomm.reconnect.enabled" value="true" />
   <property name="l2.nha.tcgroupcomm.reconnect.timeout" value="15000" />
   <!-- client-to-server reconnect -->
   <property name="l2.l1reconnect.enabled" value="true" />
   <property name="l2.l1reconnect.timeout.millis" value="15000" />
  </tc-properties>
```

Debian initszkript a Terracotta szerver indításához

- /etc/init.d/terracotta néven mentsük el root tulajdonossal a következő szkriptet, majd adjunk rá execute jogot:

```
#!/bin/sh
TC_USER=nobody
TC_GROUP=nogroup
PIDFILE=/var/run/terracotta.pid

if [`id -u` -ne 0 ](); then
        echo "You need root privileges to run this script"
        exit 1
fi

if [-f /etc/default/terracotta ](); then
    . /etc/default/terracotta
fi

if [-z "$TC_INSTALL_DIR" -o ! -d "$TC_INSTALL_DIR" ](); then
    echo "No TC_INSTALL_DIR specified or invalid TC_INSTALL_DIR"
    exit 1
fi
if [-z "$TC_CONFIG_PATH" -o ! -f "$TC_CONFIG_PATH" ](); then
    echo "No TC_CONFIG_PATH specified or invalid TC_CONFIG_PATH"
    exit 1
fi
if [-z "$NODENAME" ](); then
    echo "No NODENAME specified"
    exit 1
fi

export JAVA_HOME

JAVA_OPTS="-server -Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError $JAVA_OPTS"
TC_START_OPTS="$JAVA_OPTS \
   -Dcom.sun.management.jmxremote \
   -Dtc.install-root=$TC_INSTALL_DIR \
   -cp $TC_INSTALL_DIR/lib/tc.jar \
   com.tc.server.TCServerMain -n $NODENAME -f $TC_CONFIG_PATH"
TC_STOP_OPTS="-Dtc.install-root=$TC_INSTALL_DIR \
  -cp $TC_INSTALL_DIR/lib/tc.jar \
```

```
  com.tc.admin.TCStop -n $NODENAME"

. /lib/lsb/init-functions
. /etc/default/rcS

check_stopped () {
    return `/sbin/start-stop-daemon --test --start --pidfile "$PIDFILE" \
        --user $TC_USER --startas "$JAVA_HOME/bin/java" \
        >/dev/null`
}
start () {
    log_daemon_msg "Starting Terracotta server node ($NODENAME)..."

    if check_stopped; then

        /sbin/start-stop-daemon -S -v --make-pid --pidfile "$PIDFILE" \
            --chuid $TC_USER:$TC_GROUP --background \
            --exec $JAVA_HOME/bin/java -- $TC_START_OPTS

    else
        log_progress_msg "(already running)"
    fi
    log_end_msg 0
}
stop () {
    log_daemon_msg "Stopping Terracotta server node ($NODENAME)..."
    if $JAVA_HOME/bin/java $TC_STOP_OPTS; then
        log_progress_msg "(shutdown command sent)"
    else
        log_progress_msg "(could not send shutdown command)"
    fi
    sleep 5
    if check_stopped; then
        log_progress_msg "(cleaning persistent store)"
        rm -r /var/lib/terracotta/server/*
        log_end_msg 0
    else
        log_progress_msg "(stopping in progress)"
    fi
}
```

```
force_stop () {
    log_daemon_msg "Killing Terracotta server node ($NODENAME)..."
    /sbin/start-stop-daemon -K --pidfile $PIDFILE -x $JAVA_HOME/bin/java
}

case "$1" in
  start)
      start
      ;;
  stop)
      stop
      ;;
  force-stop)
      force_stop
      ;;
  restart)
      stop
      sleep 10
      start
      ;;
  *)
    echo "Usage terracotta start|stop|force-stop|restart"
    exit 1;;
esac

exit $?
```

- A konfiguráció az /etc/default/terracotta fájlban található:

```
NODENAME=terracotta-node-name
TC_CONFIG_PATH=/path/to/shibboleth-idp/conf/tc-config.xml
JAVA_HOME=/path/to/jvm
TC_INSTALL_DIR=/path/to/terracotta
```

Monitoring (JMX, Munin, Nagios)

- JMX: Java Management Extensions
- A Terracotta támogatja a JMX-en keresztüli monitorozást és bevatkozást
  - alapértelmezésképp jmxmp protokollon keresztül

    - másoljuk be a jmxremote_optional.jar -t a terracotta lib/ könyvtárából egy üres könyvtárba
    - indítsuk a jconsole-t a következő paranccsal: *jconsole -J-Djava.endorsed.dirs=.*
    - kapcsolódjunk a `service:jmx:jmxmp://<tc-szerver-node>:9520` url-re
  - usernév/jelszavas authentikáció esetén rmi protokollon keresztül is elérhetjük a tc szervert
- Check_jmx szkriptek nagioshoz és muninhoz

Fontosabb Terracotta MBean attribútumok

- org.terracotta:type=Terracotta Server,name=DSO
  - LiveObjectCount (int)
  - ClientLiveObjectCount (string)
- org.terracotta.internal:type=DSO Client,name=Client Transactions,subsystem=Transactions,clients=Clients,node=...
  - AvgModifiedObjectsPerTransaction (int)
  - AvgNewObjectsPerTransaction (int)
  - ObjectCreationRateBySecond (int)
  - ReadTransactionCount (int)
  - WriteTransactionCount (int)
- org.terracotta.internal:type=Terracotta Server,name=Terracotta

Server

```
* Active (bool)
* PassiveStandby (bool)
* PassiveUninitialized (bool)
* HealthStatus (String)
* State (string)
* StartTime (timestamp)
* Shutdownable (bool)
```

Nagios Terracotta szerver check

```
#!/bin/sh

TERRACOTTA_SERVER_NODES="papigw.aai.niif.hu sandbox.aai.niif.hu"
TERRACOTTA_CLIENT_COUNT=2
TERRACOTTA_SERVER_COUNT=2
JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun
```

```
TC_HOME=/usr/local/terracotta-2.7.2
TC_MONITORING=/home/hege/terracotta/terracotta-monitoring/terracotta_monitoring.jar

ACTIVE_NODES=""
STANDBY_NODES=""
CLUSTER_STATUS=""

function check_health() {
    TC_HEALTH=`echo "$1" | awk "/$2.health/{print "'$2}'`
    if [ "x$TC_HEALTH" = "xOK" ]; then
        return 0
    else
        echo TERRACOTTA CRITICAL - node $2 down
        exit 2
    fi
}
function check_active_count() {
    ACTIVE_NODES=`echo "$1" | awk '/ACTIVE-COORDINATOR/{print $1}' | sed 's/\.state://'
    ACTIVE_COUNT=`echo "$1" | awk 'BEGIN {cnt=0} /ACTIVE-COORDINATOR/{cnt++} END {print cnt}'`
    CLUSTER_STATUS="$CLUSTER_STATUS, active nodes: $ACTIVE_NODES"

    if [ "0$ACTIVE_COUNT" -eq 1 ]; then
        return 0
    else if [ "0$ACTIVE_COUNT" -gt 1 ]; then
            echo TERRACOTTA CRITICAL - multiple ACTIVE nodes $CLUSTER_STATUS
        else
            echo TERRACOTTA_CRITICAL - no ACTIVE nodes $CLUSTER_STATUS
        fi
        exit 2
    fi
}

function check_standby_count() {
    STANDBY_NODES=`echo "$1" | awk '/PASSIVE-STANDBY/{print $1}' | sed 's/\.state://'`
    STANDBY_COUNT=`echo "$1" | awk 'BEGIN {cnt=0} /PASSIVE-STANDBY/{cnt++} END {print cnt}'`
    CLUSTER_STATUS="$CLUSTER_STATUS, standby nodes: $STANDBY_NODES"

    if [ "0$STANDBY_COUNT" -eq $(($TERRACOTTA_SERVER_COUNT-1)) ]; then
        return 0
    else
```

```
        echo TERRACOTTA CRITICAL - not enough STANDBY node $CLUSTER_STATUS
        exit 2
    fi
}

function check_client_count() {
    CLIENTCOUNT=`echo "$1" | awk '/clientcount/{print $2}'`
    CLIENT_NODES=`echo "$1" | awk '/client.*address/{print $2}'`
    CLUSTER_STATUS="$CLUSTER_STATUS, client nodes: $CLIENT_NODES"
    if [ "0$CLIENTCOUNT" -eq $TERRACOTTA_CLIENT_COUNT ]; then
        return 0
    else
        echo TERRACOTTA WARNING - client count is $CLIENTCOUNT $CLUSTER_STATUS
        exit 1
    fi
}

OUTPUT=`$JAVA_HOME/bin/java -jar $TC_MONITORING $TERRACOTTA_SERVER_NODES 2>/dev/null`

check_active_count "$OUTPUT"
for i in $TERRACOTTA_SERVER_NODES; do
    check_health "$OUTPUT" "$i"
done
check_standby_count "$OUTPUT"
check_client_count "$OUTPUT"

echo TERRACOTTA OK - cluster is running $CLUSTER_STATUS

if [ "0$1" == "0--verbose" -o "0$1" == "0-v" ]; then
        echo -e "\n\n$OUTPUT"
fi
```

Munin plugin a terracotta szerver memóriahasználatának méréséhez

A check_jmx csomagban található jmx_ munin plugin mellé másoljuk oda a jmxremote_optional csomagot (sun.com-ról letölthető), majd végezzük el a következő módosítást:

```
JMXQUERY="java -cp $RDIR/jmxquery.jar:$RDIR/jmxremote_optional-1.0.1_04-b58.jar \
 org.munin.JMXQuery $SERVICE $RDIR/$CONFIGNAME"
```

Ezután a következő konfigurációt hozzuk létre terracotta_objectcount.conf néven:

```
graph_title Terracotta clustered object count
graph_category Terracotta
graph_order terracotta_objectcount

terracotta_objectcount.label cluster object count
terracotta_objectcount.jmxObjectName org.terracotta:type=Terracotta Server,name=DSO
terracotta_objectcount.jmxAttributeName LiveObjectCount
```

Majd symlinkeljük be a jmx_ szkriptet a munin pluginok közé jmx_terracotta_objectcount néven, és adjuk meg a munin-node.conf -ban a jmx hozzáférési paramétereket:

```
[jmx_*]
env.jmxurl service:jmx:jmxmp://localhost:9520
```

Tomcat monitorozása muninnal

A Tomcat JVM JMX elérésének engedélyezéséhez az /etc/default/tomcat5.5 fájlban a következő plusz kapcsolókat kell megadni:

```
CATALINA_OPTS="${CATALINA_OPTS} \
 -Dcom.sun.management.jmxremote \
 -Dcom.sun.management.jmxremote.port=8083 \
 -Dcom.sun.management.jmxremote.ssl=false \
 -Dcom.sun.management.jmxremote.authenticate=false"
```

Ezután a 8083 -as porton jmxrmi protokollon keresztül lehet elérni a menedzsment ágenst. A check_jmx pluginhoz a következő környezeti beállítást kell hozzárendelni az /etc/munin/plugin-conf.d/munin-node fájlban:

```
[jmx_catalina_*]
env.jmxurl service:jmx:rmi:///jndi/rmi://localhost:8083/jmxrmi
```

Terracotta 2.7.3 ismert hibák

Log file flood

Az L2 újracsatlakozás engedélyezése esetén egy elvesztett kapcsolat után hiába áll helyre a helyes működés, a szerver szemetel a logba:

```
2009-06-24 14:48:38,304 [ConnectionEstablisher] WARN
com.tc.net.protocol.transport.ClientMessageTransport -
ConnectionID(0.0e84473d1e744f4db1d539db88633a30): Timeout of 10000 milliseconds occured
2009-06-24 14:48:39,305 [ConnectionEstablisher] INFO
com.tc.net.protocol.transport.ClientMessageTransport -
ConnectionID(0.0e84473d1e744f4db1d539db88633a30): Attaching new connection: [...]
```

Ez egy ismert probléma: http://jira.terracotta.org/jira/browse/CDV-1252 a javítás elkészült, azonban a 2.7.3 -ba még nem került be.

Workaround-ként ki lehet kapcsolni az l2 újracsatlakozást a konfigurációban. Ilyenkor azonban rövid hálózati megszakadás is teljes node újraindítást fog okozni.

Terracotta 3.2.1 ismert hibák

Még nincs :)

Troubleshooting

- Mindig ellenőrizni kell a logfájlokat! Ha egy szerver folyton írja a logfájlját, az általában rossz jel és elárvult kliensre utal ("Could not find communication stack..." üzenet).
- A teljes klaszter újraindítása után kötelező újraindítani a klienseket (Tomcat), ugyanis a Terracotta nem fogja engedni újra csatlakozni. Ezt egyébként a szerver logfájlban jelzi is.
- Nem érdemes egyszerre indítani a két Terracotta szervert, annak könnyen összeakadás lehet a vége, ha nem tudnak dönteni egymás között.
  - ilyenkor az egyik szerver felszólítja a másik szervert a megállásra, ilyenkor azonban a diszken tárolt állapot megmarad, amit egy start parancs kiadása után nem hajlandó felhasználni a szerver processz.
  - a megoldás az initszkript 'restart' parancsa (vagy két egymás utáni start, ugyanis második kísérletre már hajlandó elindulni 'dirty' adatokkal is).
- Az `/etc/nagios/check_terracotta --verbose` parancs a teljes klaszter állapotát visszaadja (szerverek és kliensek is).
- Ha egy probléma nem oldható meg csak az egyik szerver és a kliensek újraindításával, akkor a teljes klasztert újra kell indítani: mindkét szerver leállítása és a perzisztens adatok

gondos törlése után egyesével újraindíthatóak a szerverek majd az aktív szerver indulása után a kliensek (Tomcat).

Kliens library

- a Tomcat webkonténerben fut
- a `/var/log/terracotta/client/log*/terracotta-client.log` fájlba logol
- **JVM váltáskor, frissítéskor újra kell generálni!** Ezt a `/usr/local/terracotta/bin/make-boot-jar.sh` szkripttel lehet megtenni, előtte törölni kell a `/usr/local/terracotta/lib/dso-boot` könyvtár tartalmát.

Szerver processz

- külön processzként fut
- a `/var/log/terracotta/server/log*/terracotta-server.log` fájlba logol
- a `/var/lib/terracotta/server/` könyvtárban tárol saját adatokat, amit kézzel történő tiszta újraindítás előtt törölni kell

Nagios riasztások és megoldásuk

- Túl kevés a kliens (client count is xxx)
  - **OK**: a Tomcat kliens megállt vagy megszakadt a kommunikáció a szerverrel.
  - **Megoldás**: a kliens listában nem szereplő Tomcat-et újra kell indítani.
- Nincs passzív node (not enough passive node)
  - **OK**: az egyik Terracotta szerver épp adatot szinkronizál a másiktól és ezért `passive-uninitialized` állapotban van.
  - **Megoldás**: pár percet érdemes várni, amíg a szinkronizáció befejeződik. Ha nem oldódik meg a probléma magától, akkor újra kell indítani a Terracotta szerver processzt.
- Nem elérhető egy node (node xxx is down)
  - **OK**: az egyik Terracotta szerver nem elérhető.
  - **Megoldás**: újra kell indítani.

Adminisztrációs feladatok

JVM frissítése

A következő leírás meglehetősen az NIIF Intézet saját infrastruktúrájára specifikus, de talán más környezetekben is felhasználható.

Lépések összefoglalása

1. Terheléselosztóból a frissítendő node-ot kivenni
2. Tomcat, Terracotta leállítása a frissítendő node-on
3. JVM beállítások (`/etc/java-6-openjdk/security`, ill. esetleg `/usr/lib/jvm/java-6-openjdk/jre/lib/security`) elmentése. Ez fontos azért, mert bizonyos JVM upgrade-ek (legalábbis a múltban) felülírták a tanúsítvány tárat, és ez nehezen javítható hibához vezetett (pl. az LDAP-hoz nem tudott kapcsolódni az IdP)
4. JVM frissítés
5. Boot jar generálás
6. (Ha megváltozott a jar): Tomcat konfigban a boot jar átírása az újra
7. Ellenőrzés, hogy megváltozott-e a cacerts ($JAVA_HOME/lib/security/cacerts). Ha igen, akkor írjuk felül az elmentett változattal
8. Terracotta, **majd utána** Tomcat indítás
9. (Az LVS magától visszateszi a megjavuló klaszter node-ot, de erről nem árt meggyőződni)

Shell parancsok

```
ldir2:~# ipvsadm -d -t idp.niif.hu:8443 -r idp1.aai.niif.hu:8443
ldir2:~# ipvsadm -d -t idp.niif.hu:https -r idp1.aai.niif.hu:https
ldir2:~# watch "ipvsadm -L -t idp.niif.hu:https  && ipvsadm -L -t idp.niif.hu:8443"


idp1:~$ sudo /etc/init.d/tomcat6 stop
idp1:~$ sudo /etc/init.d/terracotta stop
idp1:~$ tar czf security.tgz /etc/java-6-openjdk/security
idp1:~$ sudo aptitude safe-upgrade

idp1:~$ sudo env JAVA_HOME=/usr/lib/jvm/java-6-openjdk
/usr/local/terracotta/platform/bin/make-boot-jar.sh \
-f /etc/shibboleth-idp/tc-config.xml
idp1:~$ sudo vim /etc/default/tomcat6

idp1:~$ sudo /etc/init.d/terracotta start
idp1:~$ sudo /etc/init.d/tomcat6 start
```