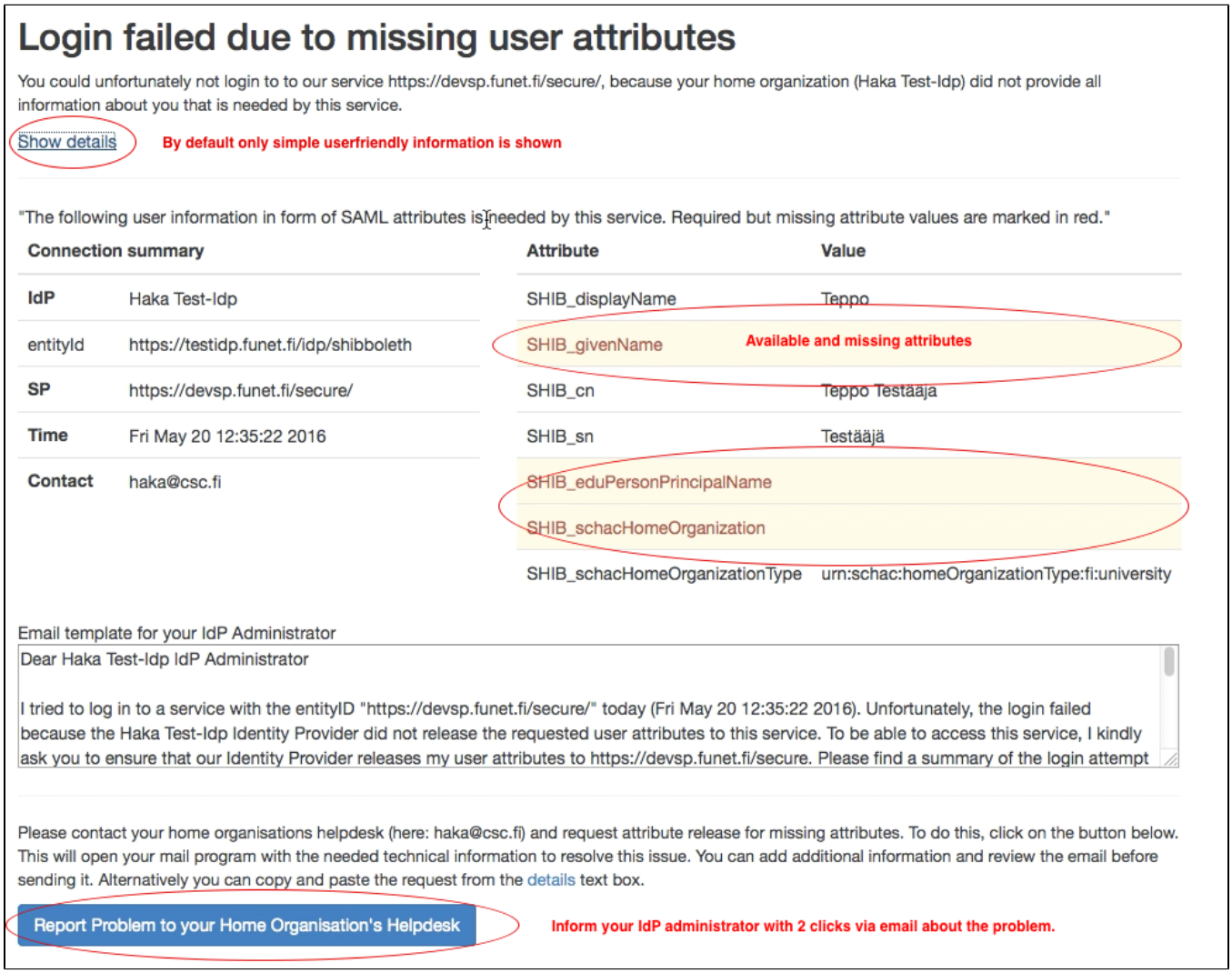
How to configure Shibboleth SP attribute checker

Shibboleth SP attribute checker

The list of attributes released by an IdP varies heavily and occasionally required attributes needed by an SP are not released by the user's IdP. This leads to failed logins and the error page doesn't give clear information of the failure reason (eg. what attributes are missing). You can always check the attributes on an application level. The approach described here is done on the Shibboleth SP level (requires Shibboleth 2.5+) and does not require changes to the application using the attibutes.

Shibboleth SP provides a hook for performing attribute checks for required attributes and a attribute extractor for fetching IdP metadata attributes where the login was performed. The end result (from the user's perspective) then is an error message like shown below in case the user's Identity Provider did not release sufficiently user information to the service:



One also finds some further explanation and a quick demo in this screen cast.

Attribute Checker Handler

The AttributeChecker validates the user session against attributes specified as a required. If requirements are fulfilled, the login completes otherwise an error page is displayed instead. Note that the required attributes have to be "hard coded" here and kept in sync with the required attributes expressed in the Metadata.

Configuration

Add a sessionHook for attribute checker: sessionHook="/Shibboleth.sso/AttrChecker" to ApplicationDefaults. Also add the metadataAttributePrefix="Meta-" (This will be explained later).

In context: /etc/shibboleth/shibboleth2.xml -> ApplicationDefaults element

```
<ApplicationDefaults entityID="https://<HOST>/shibboleth"
  REMOTE_USER="eppn persistent-id targeted-id"
  signing="front" encryption="false"
  sessionHook="/Shibboleth.sso/AttrChecker"
  metadataAttributePrefix="Meta-" >
```

Add the attribute checker handler with the list of required attributes to Sessions (in the example below: eppn, displayName).

/etc/shibboleth/shibboleth2.xml -> Sessions element

```
<Handler type="AttributeChecker" Location="/AttrChecker" template="attrChecker.html" attributes="eppn
displayName" flushSession="true"/>
```

If you want to describe more complex scenarios with required attributes, operators such as "AND" and "OR" are available.

```
<Handler type="AttributeChecker" Location="/AttrChecker" template="attrChecker.html" flushSession="true">
 <OR>
   <Rule require="displayName"/>
   <AND>
     <Rule require="givenName"/>
     <Rule require="surname"/>
   </AND>
 </OR>
</Handler>
```

Now we have an session hook for the attribute checker to check specified attributes before a user login is completed. For customization of the error page (attrChecker.html) we want to enable the "Attribute Extractor" with the type "metadata" to be able to fetch IdP attributes from the metadata feed. The Attribute we need is the email address of the IdP support contact. We already added metadataAttributePrefix to the ApplicationDefaults element.

Add the AttributeExtractor element of the type metadata next to the already existing type XML: (<AttributeExtractor type="XML" validate="true" path="attribute-map.xml"/>)

/etc/shibboleth/shibboleth2.xml -> ApplicationDefaults element

```
<!-- Extracts support information for IdP from its metadata. -->
<AttributeExtractor type="Metadata" errorURL="errorURL" DisplayName="displayName"
                    InformationURL="informationURL" PrivacyStatementURL="privacyStatementURL"
                    OrganizationURL="organizationURL">
  <ContactPerson id="Support-Contact"  contactType="support" formatter="$EmailAddress" />
  <Logo id="Small-Logo" height="16" width="16" formatter="$_string"/>
</AttributeExtractor>
```

When you modify shibboleth2.xml you can test validity of the configuration file with command "shibd -t". If configuration file is still valid XML you can now restart your shibboleth with "sudo service shibd restart". Shibboleth should anyways reload configuration file if it detects any change on it.

Logging

Shibboleth SP doesn't track nor log failed logins due to missing attributes. The Shibboleth SP web server can be used for "pixel tracking". This means that you load an image (eg: containing only one transparent pixel) from the web server from where you can monitor logs and observe access for you image. In the url of your image you can also insert details you want to see, eg: Authentication source (IdP) and missing attributes.

Replace the image with your existing one from the following code or comment it out if you dont need it. Example below loads track.png from document root and adds variables like "idp" containing the entityID of the authentication source and "miss" denoting missing attributes.

Pixel tracking

```
<img title="track" src="/track.png?idp=<shibmlp entityID/>&miss=<shibmlpifnot displayName>-displayName<
/shibmlpifnot><shibmlpifnot givenName>-givenName</shibmlpifnot><shibmlpifnot cn>-cn</shibmlpifnot><shibmlpifnot
sn>-sn</shibmlpifnot><shibmlpifnot eduPersonPrinpalName>-eduPersonPrinpalName</shibmlpifnot><shibmlpifnot
schacHomeOrganization>-schacHomeOrganization</shibmlpifnot><shibmlpifnot schacHomeOrganizationType>-
schacHomeOrganizationType</shibmlpifnot>" alt="" width="1" height="1" />
```

Template customization

The attrChecker.html is located in the "/etc/shibboleth" directory. If you don't want to edit it by yourself, you can use the ready made template. The template has links to external components such as jquery and bootstrap. They are fetched on the fly from third party sources. Basically there are three locations needing modifications:

- The pixel tracking link after the comment "PixelTracking". The Image tag and all required attributes after the variable must be configured here. After "miss=" define all required attributes you updated in shibboleth2.xml using shibboleth tagging. Eg <shibmlpifnot $attribute>-$attribute< /shibmlpifnot> (this echoes $attribute if it's not received by shibboleth). This example uses "-" as a delimiter.

- The table showing missing attributes between the tags "TableStart" and "TableEnd". You have to insert again all the same attributes as above.

Define row for each required attribute (eg: displayName below)

```
<tr <shibmlpifnot displayName> class='warning text-danger'</shibmlpifnot>>
  <td>displayName</td>
  <td><shibmlp displayName /></td>
</tr>
```

- The email template between the tags "<textarea>" and "</textarea>". After "The attributes that were not released to the service are:". Again define all required attributes using shibboleth tagging like in section 1 (eg: <shibmlpifnot $attribute> * $attribute</shibmlpifnot>). Note that for SP identifier target URL is used instead of entityID. There arent yet any tag for SP entityID so you can replace this target URL manually.

You can also update attrChecker.html with a Perl-script (attrChecker.pl). The script extracts the required attributes from the Attribute Checker handler element in shibboleth2.xml and modifies attrChecker.html accordingly (Note that script doesnt work with complex scenarios using AND and OR operators, it uses only "attributes" attribute from the handler). If you customize attrChecker.html and execute the Perl-script, make a backup of attrChecker.html before executing attrChecker.pl. If the script doesn't find the tags it needs for replacing content, it might break the template. The script updates the PixelTracking link by replacing shibboleth tags between miss= and following ", attribute table rows between "TableStart" and "TableEnd" and after line "The attributes that were not released to the service are:" until the next empty line.

- attrChecker.html and attrChecker.pl script can be downloaded from the GitHub