Authentication and Authorisation for Research and Collaboration

SAML and Shibboleth SP Introduction

Training by AARC

Agenda - What we will learn

1. Short intro to Federations, SAML and Shibboleth (~30m)
2. Installation and setup of Shibboleth SP (~30m)
3. Demo with Federation Integration (~15m)
4. Demo with Proxy Integration (~10m)
5. Q&A (~15m)

Federated authentication

Collaboration between different organizations is possible by either asking people to create an account in each organization or by enabling **federated access.**

Organizations run an **Identity Provider** (IDP), which

- Manages identity information of its users
- “Offers authentication as a service”

Organizations run a **Service Provider** (SP), which

- Protects some application in the background
- Relies on the identity information received by an IDP

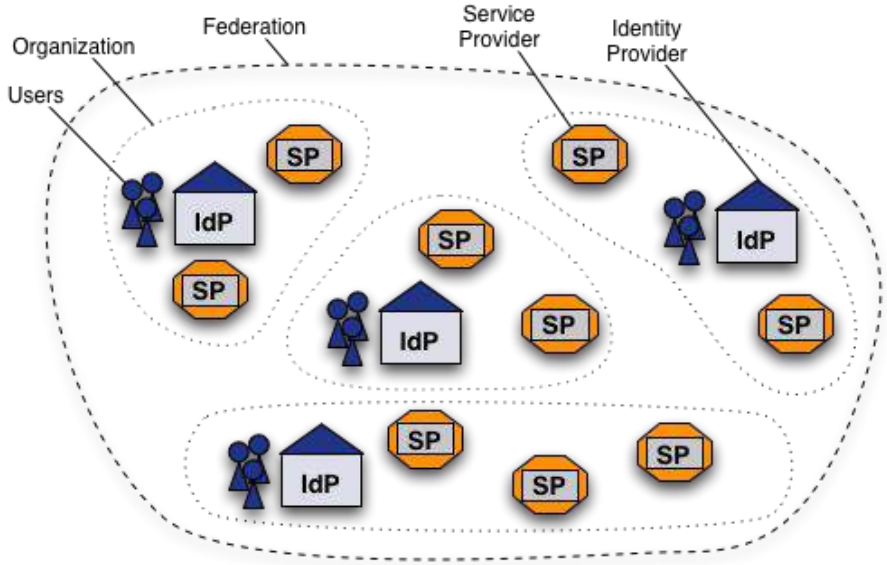
Advantages include **less account-management**, **central identity sources** and **Single Sign-On** (SSO).

What is a Federation

A federation is a **collection of organizations** that agree to interoperate under a certain rule set. Federations will usually define trusted roots, authorities and attributes, along with distribution of metadata representing this information.

In general each organization participating in a federation operates:

- one **Identity Provider** (IdP) for their users, and
- any number of **Service Providers** (SP) or applications.



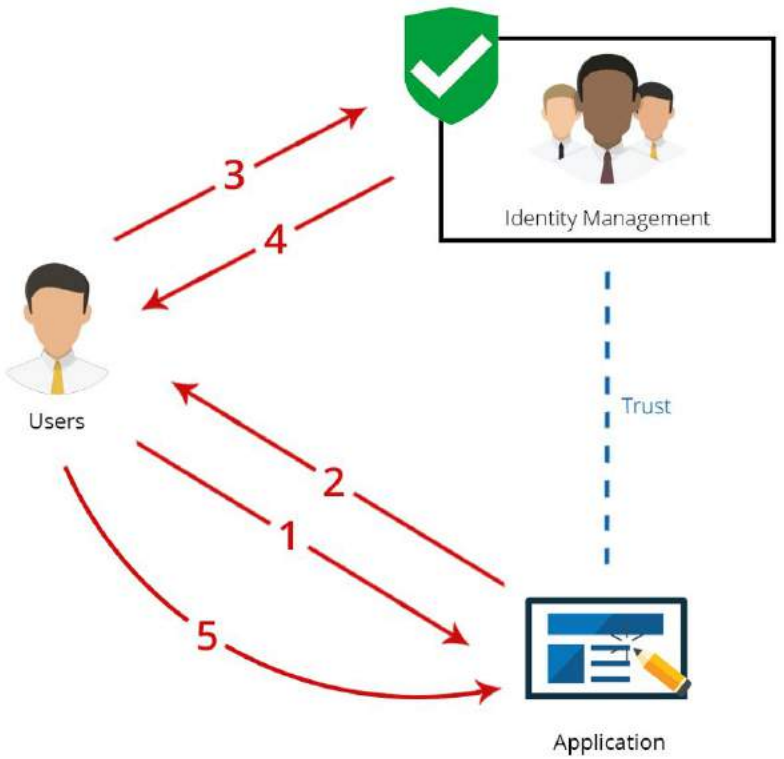
How to federate an application

To federate an application, specific library or software must be configured to handle federated authentications:

- for **SAML**: e.g. **Shibboleth SP** or **SimpleSAMLphp**
- for **OIDC**: e.g. **pyoidc** library

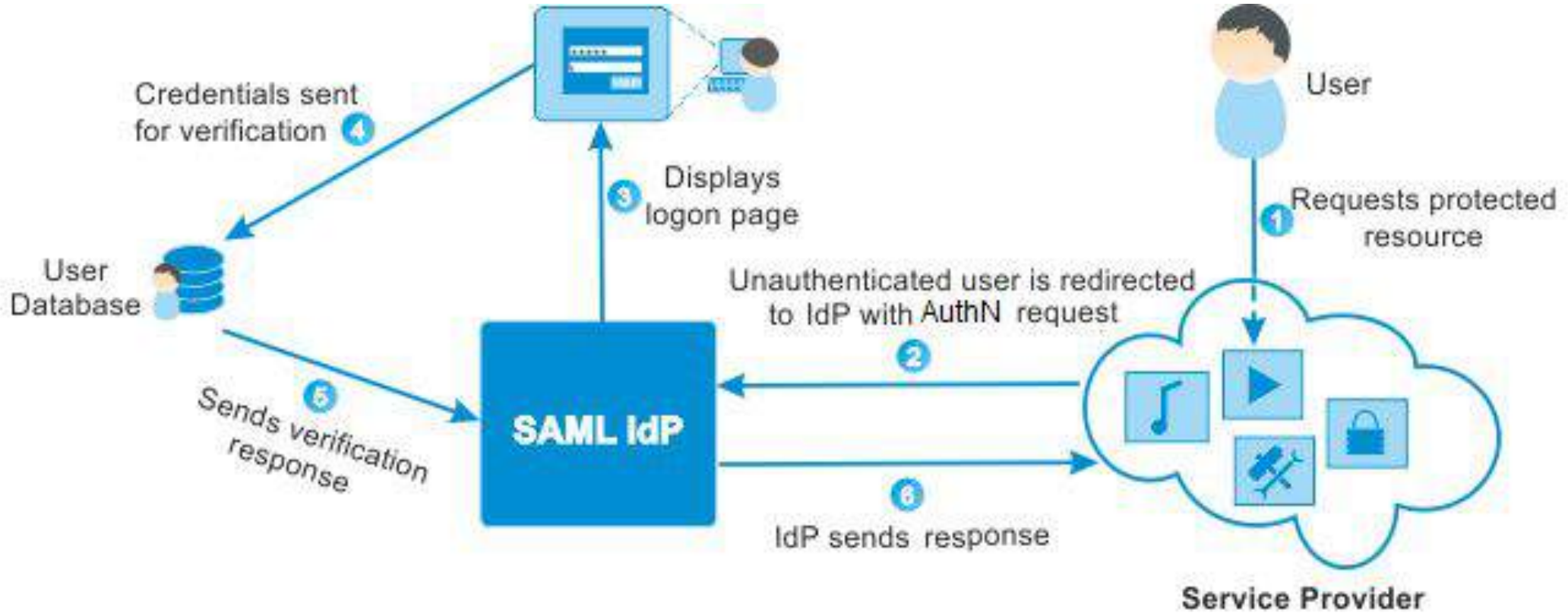
All these software takes the responsibility to interact with entities at the federation level (=IdPs) to exchange digital identities information.

The application itself can then consume the user attributes obtained from an IdP.



How to deal with authentication

Example of the authentication flow with SAML:



Trust between IdPs and SPs

Trust between IdPs and SPs is established with **technical and policy** means.

IdPs and SPs usually use **PKI** to **communicate in a secure way**. They tend to use:

- **encryption**: to prevent someone unauthorized from reading the messages;
- **signature**: to guarantee sender and receiver are exactly the entities they pretend to be *(avoid man in the middle attack)*

Usually this information needs to be configured beforehand (e.g. in SAML: SP and IdP explicitly add each other's **Metadata** and create trust that way)

This might work when you have one SP and only a couple of IdPs. Do you want to offer a service to all IdPs in europe? Good luck...

Trust between IdPs and SPs in federations

The **trust** between IdPs and SPs is very commonly **created by federations** that:

- accept **registration** for all entities in the federation
- **verify** their adherence to federation policies
- create and distribute a set of **trusted metadata** to all entities willing to operate in the federation.

Every entity in the federation trusts the federation operator and consumes the combined metadata of all participants

Discovery Service

Since there are several IdPs in the combined metadata of the federation, the SP needs to choose which IdP to use

This is done by a Discovery Service (DS), which is commonly run by the federation operator or configured directly on the SP as an embedded and independent solution, that presents to user a list of IdPs he can select to go ahead with the resource access.

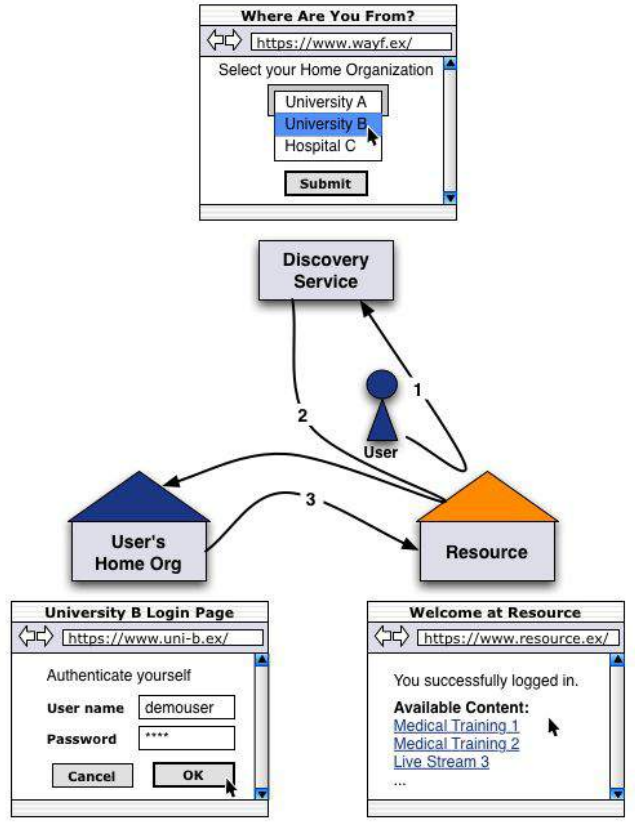
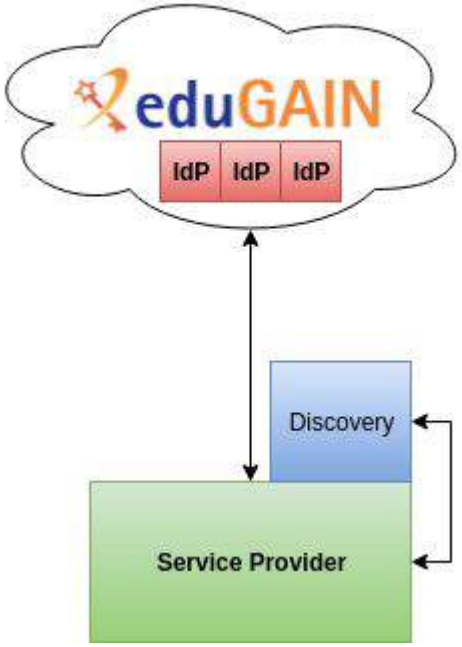


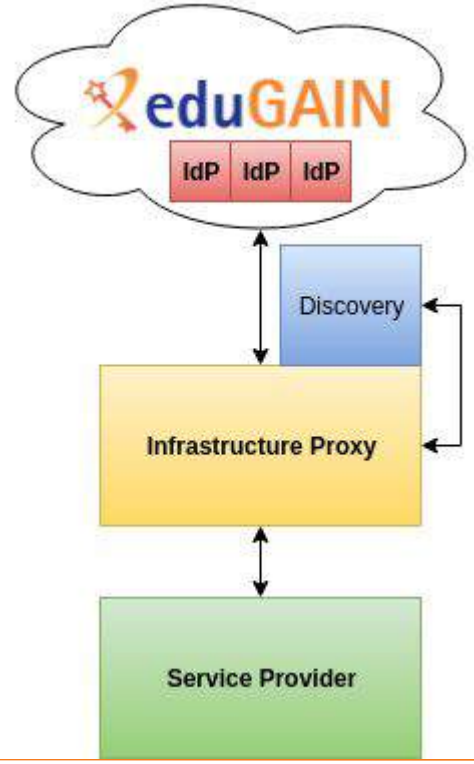
Image source: switch.ch

Two approaches to consider...

- It is common that several services in a federation can be grouped (e.g. they belong to a research community with very similar requirements)
- Connect directly to a federation (e.g. EduGain)



- Connect to a Infrastructure Proxy



Recap

Federated authentication allows collaboration of multiple organizations without the need to have a separate account at every organization.

Federations manage participating organizations and create a set of metadata to create trust between participants.

SSO protocols are used to convey information between entities (IdPs and SPs) in a federation, SAML2 is one such protocol.

Shibboleth

*„Shibboleth is a **standards based**, **open source** software package for **web single sign-on** across or within organizational boundaries.“*

- Began as an Internet2 activity in 2000
- SAML (2.0)
- Maintained by the Shibboleth Consortium
- Widespread usage in research & education
- Several products
  - Identity Provider (IdP)
  - Service Provider (SP)
  - Discovery Service etc.

Shibboleth Service Provider (SP)

- Implements the SP component in a SAML communication:
  - i.e. protects a web application behind it and deals with SAML communication
- There are two parts:
  - A background service or "daemon" process (shibd)
    - Keeps states, evaluates protocol messages
  - A Webserver module (mod_shib)
    - Protects Locations/Directories, defines Access Rules
- Current release: V3.0.2
- Available through:
  - Binaries for Windows, OSX, RPM-based Linux
  - APT Repository for Ubuntu/Debian (2.x)

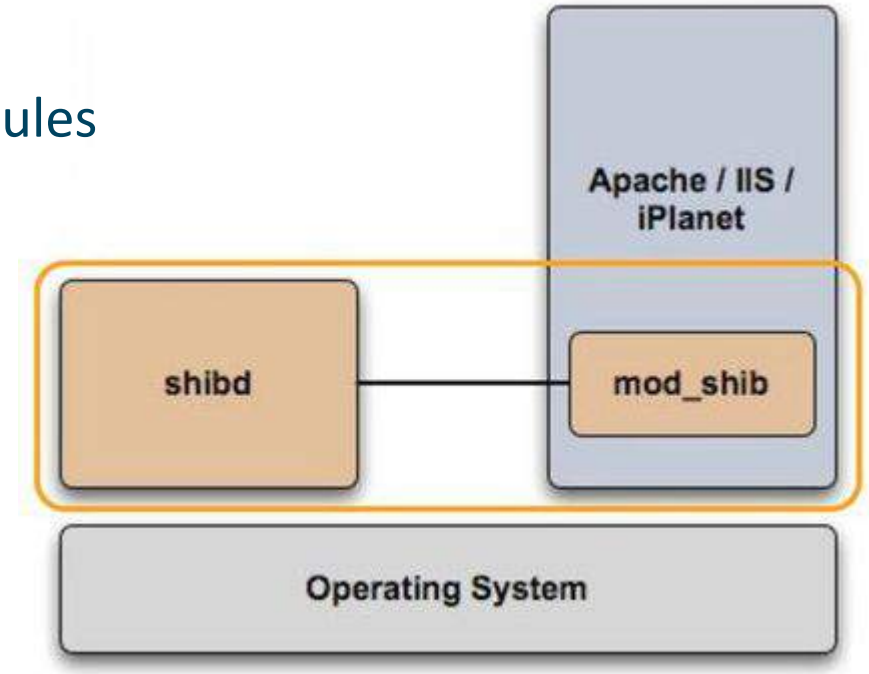


Image source: switch.ch

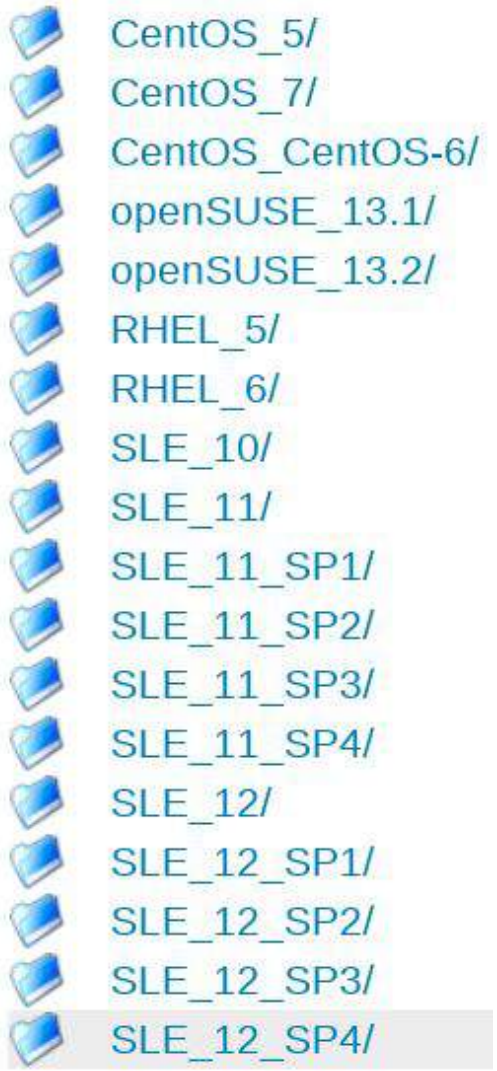
Shibboleth SP Installation

- Debian/Ubuntu distros:

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

- RPM-based distros:

```
http://download.opensuse.org/repositories/security://shibboleth
```



Configuration

- Main configuration file: /etc/shibboleth/shibboleth2.xml
- <ApplicationDefaults>: a “container” for the SP common configurations
- <Sessions>: how to manage SSO process (e.g. ssl, duration)
  - <SessionInitiator> defines how the SP should get a session (some shorthands for e.g. <SSO> and <Logout> exist)
  - <Handler> provides endpoints for different functionalities (/Metadata, /Session, /Status, ...)

```
<ApplicationDefaults entityID="https://example.org/shibboleth"
    REMOTE_USER="eppn persistent-id targeted-id">

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

    <SSO discoveryProtocol="SAMLDS"
        discoveryURL="https://example.org/ds">
      SAML2 SAML1
    </SSO>

    <Handler type="MetadataGenerator" Location="/Metadata"
        signing="false"/>

    <Handler type="Status" Location="/Status" acl="127.0.0.1 ::1"/>

    <Handler type="Session" Location="/Session"
        showAttributeValues="false"/>

  </Sessions>
```

Configuration cont'd

- <MetadataProvider> (static or dynamic)
  - e.g. Federation Metadata…
  - … or single metadata file from Proxy
- AttributesExtractor & AttributesFilter configure how the SP should provides attributes to applications
- CredentialsResolver needed to sign SP messages and encrypt messages for IdPs (the certificate needs to be published in the metadata)
- <ApplicationOverrides> only if multiple logical SPs run on a single installation

```
<MetadataProvider type="XML" validate="true"
    uri="http://example.org/fed-metadata.xml"
    backingFilePath="fed-metadata.xml" reloadInterval="7200">

    <MetadataFilter type="Signature"
        certificate="md_federation.pem"/>
</MetadataProvider>

<AttributeExtractor type="XML" validate="true"
    reloadChanges="false" path="attribute-map.xml"/>

<AttributeFilter type="XML" validate="true"
    path="attribute-policy.xml"/>

<CredentialResolver type="File"
    key="sp-key.pem" certificate="sp-cert.pem"/>

<!--<ApplicationOverride /> -->

</ApplicationDefaults>
```

Demo: Installation and basic configuration (Debian example) - Part 1

1) Install software required:
- `sudo apt install apache2 libapache2-mod-shib2 ntp openssl --no-install-recommends`

2) Create SP metadata credentials (30 years valid) with the entityID into the certificate SAN:
- `sudo /usr/sbin/shib-keygen -y 30 -e https://sp.example.org/shibboleth`

3) Activate attributes support for the SP by removing comments on '`attribute-map.xml`'

4) Edit SP configuration:
- `sudo vim /etc/shibboleth2.xml`
  - **<ApplicationDefaults>** ⇒ entityID & REMOTE_USER
  - **<Sessions>** ⇒ checkAddress + handlerSSL + cookieProps
  - **<SSO>** ⇒ Remove SAML1

Demo: Installation and basic configuration (Debian example) - Part 2

5) Restart “shibd” daemon to apply all changes:

- `sudo systemctl restart shibd.service`

6) Enable Apache2 “mod_shib” module:

- `sudo a2enmod shib2`
- `systemctl reload apache2.service`

Connect Shibboleth SP to a federation (IDEM GARR AAI Example)

1. Register the SP Metadata on the Identity Federation:
   - Download your SP metadata from **`https://sp.example.org/Shibboleth.sso/Metadata`**
     ...and share it with the Identity Federation (by email, web application/entity registry, …)
2. Set up the Discovery Service for our SSO (**`shibboleth2.xml`**):

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://wayf.idem-test.garr.it/WAYF">
  SAML2
</SSO>
```

3. Set up MetadataProvider to retrieve the Federation metadata (**`shibboleth2.xml`**):

```
<MetadataProvider type="XML" uri="http://www.garr.it/idem-metadata/idem-test-metadata-sha256.xml"
legacyOrgName="true" backingFilePath="idem-test-metadata-sha256.xml" reloadInterval="600">
    <MetadataFilter type="Signature" certificate="idem_signer.pem"/>
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="864000" />
</MetadataProvider>
```

4. Retrieve Federation metadata signing certificate (needed to validate it):
   - `sudo wget https://www.idem.garr.it/documenti/doc_download/321-idem-metadata-signer-2019 -O /etc/shibboleth/idem_signer.pem`
5. Check Shibboleth SP configuration and restart the service to apply changes:
   - `sudo shibd -t ; sudo systemctl restart shibd.service`

Demo: Interaction with Federation

1. Define attributes required by the application and inform the federation.
2. Develop the federated web application using the User Session
3. Protect application through Apache2 “mod_shib” module
4. Wait that all Identity Providers(IdP) of the federation received the metadata update
5. Try access to the federated resource with a test IdP

How to protect PHP application and use attributes from the session

The example below shows a “secure” federated application using attribute “mail” provided by the IdP:

1. Require Shibboleth authentication but don't limit who can authenticate (`/etc/apache2/conf-enabled/secure.conf`)

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

2. Use the variables collected in the Browser Session as follow:

```
<html>
 <head> <title>Example Federated PHP Application</title> </head>
 <body>
  <?php
   //The REMOTE_USER variable holds the name of the user authenticated by the web server.
   print "<p>Your REMOTE_USER is <strong>" . $_SERVER["REMOTE_USER"] . "</strong></p>";
   print "<p>Your email is <strong>" . $_SERVER['mail'] . "</strong></p>";
  ?>
 </body>
</html>
```

Connect Shibboleth SP to a single IdP (e.g. Infrastructure Proxy)

1. Exchange the SP Metadata with the IdP:
   - Download your SP metadata from **`https://sp.example.org/Shibboleth.sso/Metadata`**
     `...`and send it to the recipient IdP (by email, web application/entity registry, chat, …)

2. Set up Shibboleth SP to use the Proxy (`shibboleth2.xml`), since it's just one IdP, no DS needed:

```
<SSO entityID="https://aaiproxy-integration.de.dariah.eu/idp">
  SAML2
</SSO>
```

3. Set up Shibboleth SP to retrieve correctly Proxy Metadata (`shibboleth2.xml`):

```
<MetadataProvider type="XML"
uri="https://aaiproxy-integration.de.dariah.eu/dariah-proxy-idp-integration.xml" legacyOrgName="true"
backingFilePath="dariah-proxy-metadata.xml" reloadInterval="600" />
```

4. Check Shibboleth SP configuration and restart the service to apply changes:
   - `sudo shibd -t ; sudo systemctl restart shibd.service`

Useful Links

- HOWTO Install and Configure a Shibboleth SP v2.x on Debian Linux 9 (Stretch) - The Italian Federation example
- REFEDS Discovery Guidelines
- How to configure Shibboleth SP attribute checker - eduGAIN Wiki

Thank you

Any Questions?

https://aarc-project.eu

The work leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 730941 (AARC2).