Shibboleth Service Provider Hands-on Training

Installing and Configuring a Shibboleth 2 Service Provider

Authentication and Authorisation for Research and Collaboration

Credits and General Information

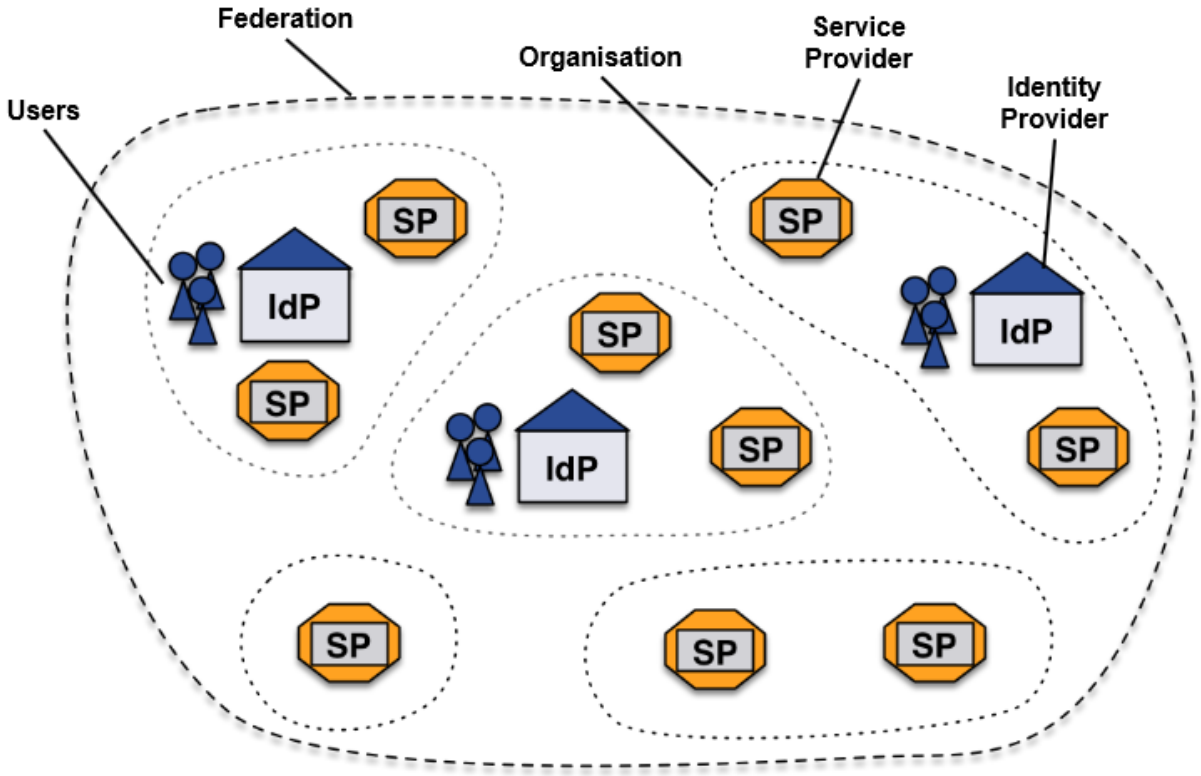
- Slides were originally created by Scott Cantor (Internet2 Developer of the Shibboleth Service Provider) and SWITCHaai (https://www.switch.ch/aai/)
- Course material is adapted for use in AARC Service Provider training
- If you see this  on a slide, hands-on work is required
- URLs at the bottom right of the sides point to more details

Federated Identity Management

- Federated Identity Management (FIM) securely shares information managed at a user's home organisation with remote services:
  - Within FIM systems, it doesn't matter if the service is in your administrative domain or another, it is all handled the same
- In Federated Identity Management:
  - Authentication (AuthN) takes place where the user is known
    - An Identity Provider (IdP) publishes authentication and identity information about its users
  - Authorisation (AuthZ) happens on the service side
    - A Service Provider (SP) relies on the AuthN at the IdP, consumes the information the IdP provided and makes it available to the application
  - An entity is a generic term for IdP or SP
- The first principle within federated identity management is the active protection of user information:
  - Protect the user's credentials
  - Protect the user's personal data, including the identifier

Federation

- A group of organisations running IdPs and SPs that agree on a common set of rules and standards:
  - It is a label used to talk about a collection of organisations
  - An organisation may belong to more than one federation at a time
- The grouping can be on a regional level or on a smaller scale (e.g. large campus)



Federation Metadata

An XML document that describes a federation

- Contains:
  - Unique identifier for each entity, known as the **entityID**
  - Endpoints where each entity can be contacted
  - Certificates used for signing and encrypting data
- May contain:
  - Organisation and person contact information
  - Information about which attributes an SP wants/needs
- Metadata is usually distributed by a public HTTP URL:
  - The metadata should be digitally signed
  - Signature should be verified!
  - Bilateral metadata exchange scales very badly
- Metadata must be kept up to date, so that:
  - New entities can interoperate with existing ones
  - Old or revoked entities are blocked

Benefits of Federated Identity Management

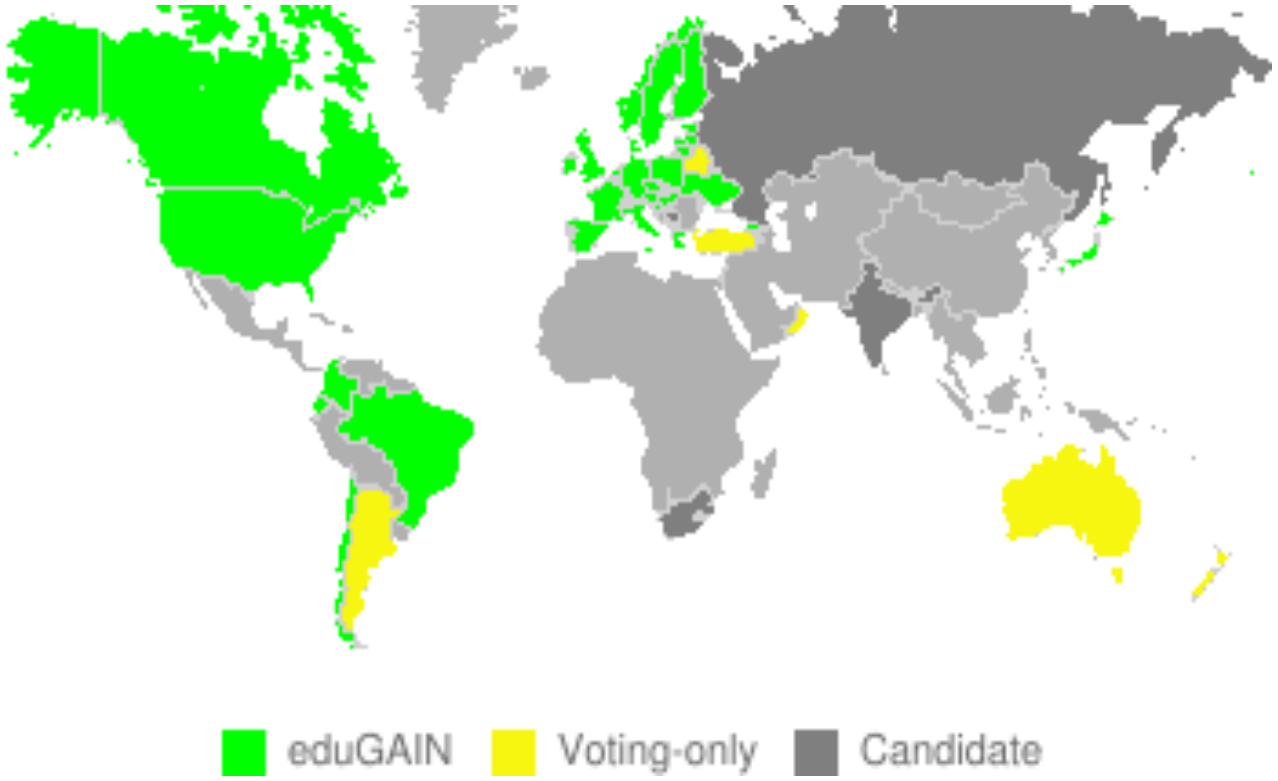
- Reduces work
  - Authentication-related calls to Penn State University's helpdesk dropped by 85% after they installed Shibboleth
- Provides current data
  - Studies of applications that maintain user data show that the majority of data is out of date
  - Are you "protecting" your app with stale data?
- Insulation from service compromises
  - Data gets pushed to services as needed
  - An attacker can't get everyone's data on a compromised server
- Minimise attack surface area
  - Only the IdP needs to be able to contact user data stores
  - All effort can be focused on securing this single connection instead of one (or more) connections per service

Other Gains from Federated Identity Management

- Users generally find the resulting single sign-on experience to be nicer than logging in numerous times
- Usability-focused individuals like that the authentication process is consistent, regardless of the service accessed
- A properly maintained federation drastically simplifies the process of integrating new services

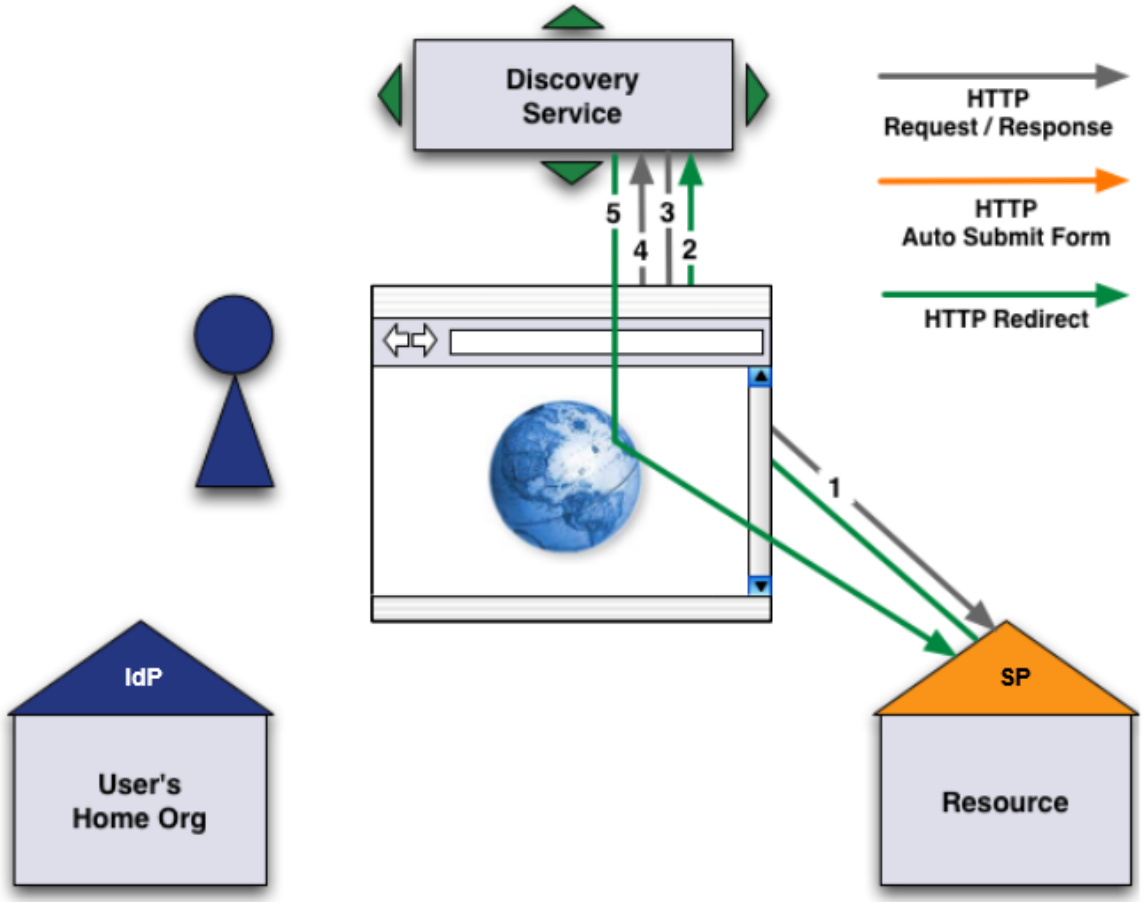
eduGAIN Interfederation

- Users get access to services from other federations
- eduGAIN is the GÉANT Interfederation Service
- provides an efficient, flexible way for participating federations, and their affiliated users and services, to interconnect
- http://services.geant.net/edugain



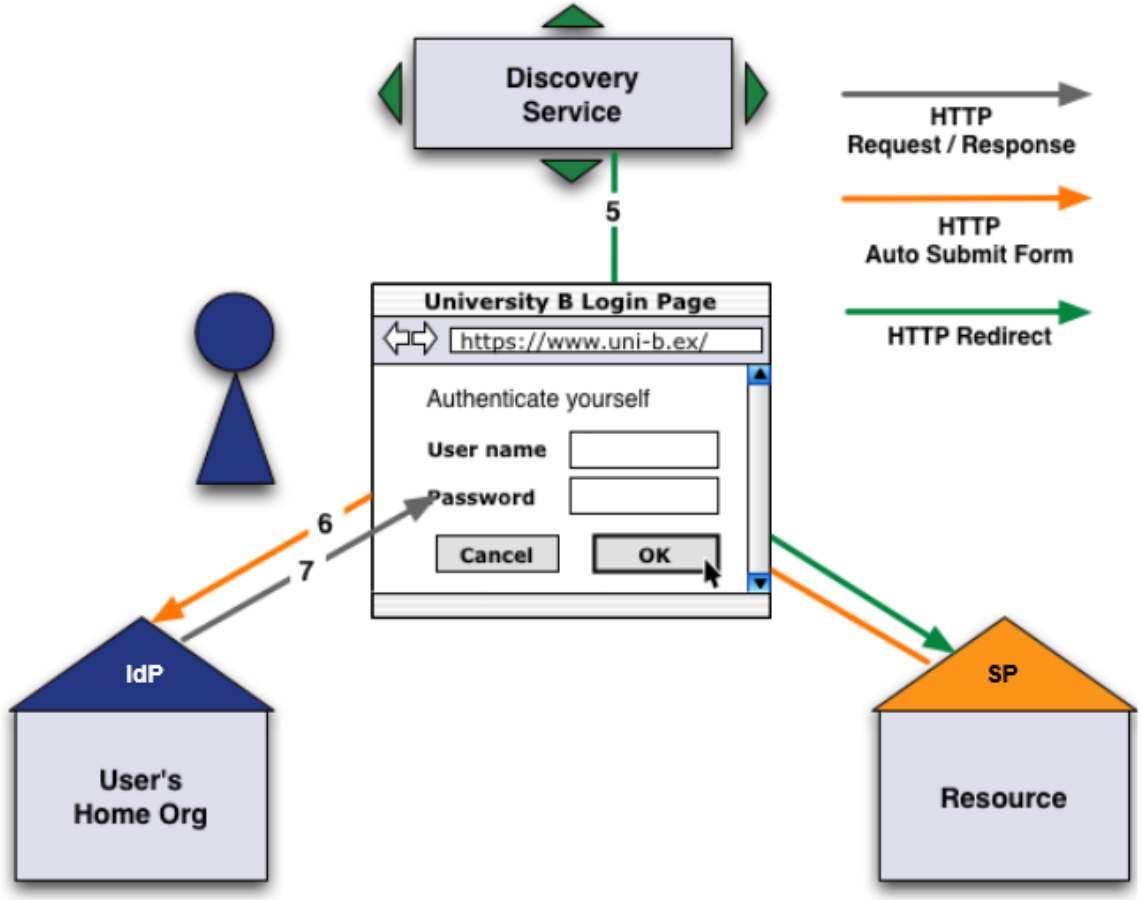
SAML2 Protocol Flow, Web Browser SSO, Phase 1

1. The user opens a web browser and accesses the Service Provider
2. The user is redirected to the Discovery Service by the Service Provider
   Consequently, the web browser sends a new request to the Discovery Service
3. The Discovery Service responds with the web page that allows the user to select an Identity Provider
4. On the Discovery Service page, the user submits the Identity Provider selection
5. The Discovery Service sends a redirect to the SP return destination, including the IdP selection



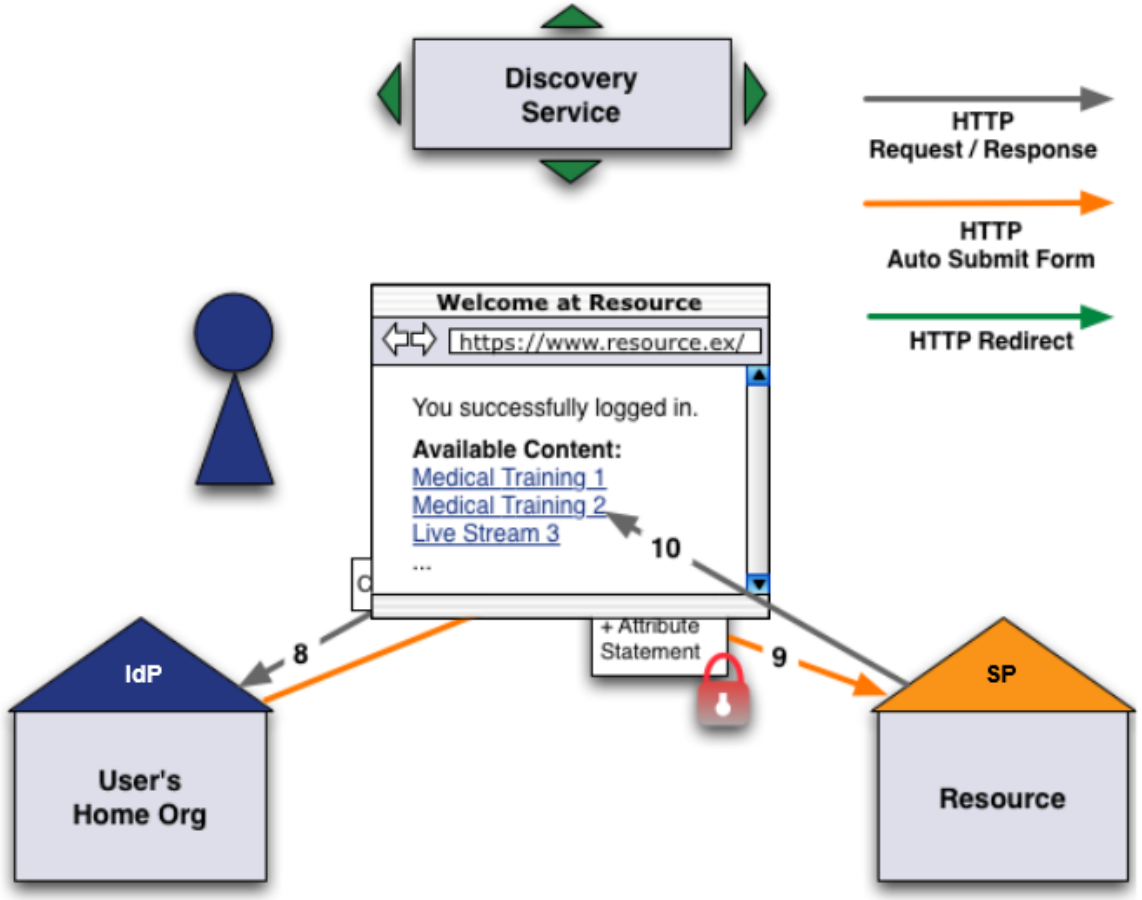
SAML2 Protocol Flow, Web Browser SSO, Phase 2

5. The browser is redirected to the Service Provider by the Discovery Service
6. The session initiator of the Service Provider creates an authentication request and returns it within an auto-submit-post-form to the browser. The browser posts the SAML AuthN Request automatically to the Identity Provider using JavaScript
7. The Identity Provider checks the authentication request. Because the user hasn't yet been authenticated, the Identity Provider sends a redirect to the appropriate login page (usually: Username/Password)



SAML2 Protocol Flow, Web Browser SSO, Phase 3

8. The user types his username and password credentials and submits them to the Identity Provider
9. The Identity Provider verifies the credentials. If authentication succeeds, the IdP issues an assertion for the SP and returns it within an auto-submit-post-form to the browser. The web browser immediately posts the SAML Assertion to the Service Provider with use of JavaScript automatically. The Service Provider processes the SAML assertion including the authentication and attribute statements
10. Finally, the Service Provider starts a new session for the user and redirects the user to the previously requested resource. Now, the user is authenticated and gains access to the resource, depending on the access rules configured for the resource

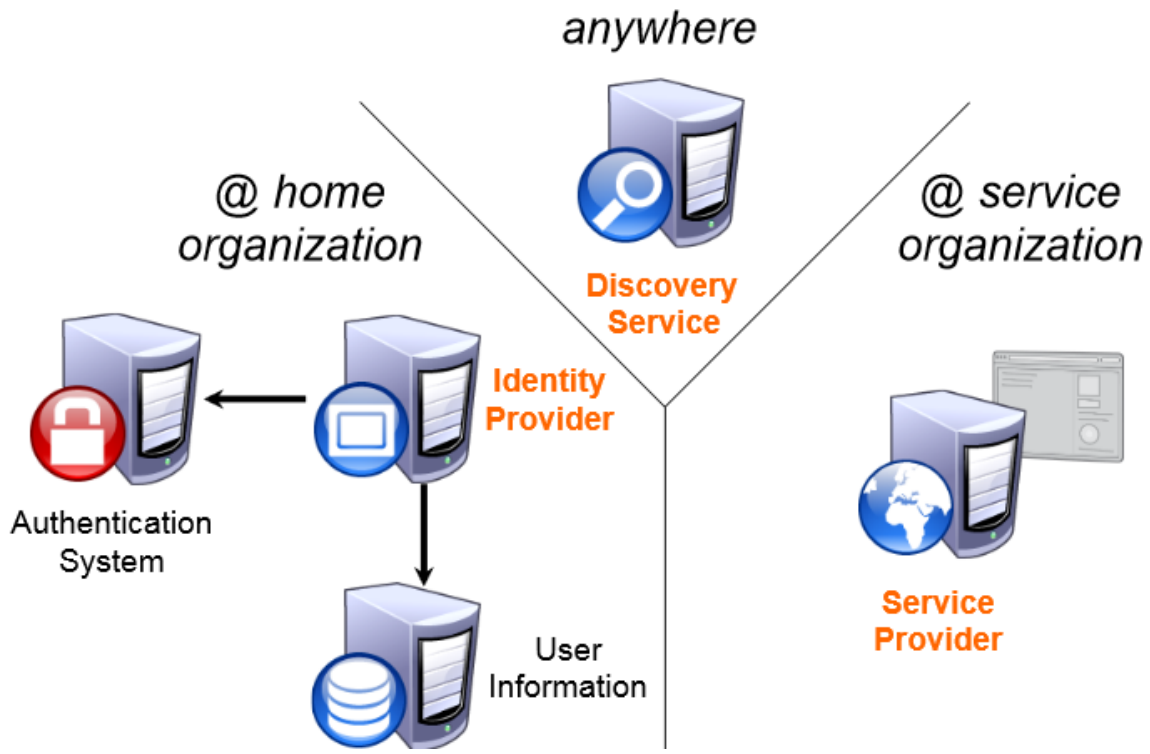


Introduction to Shibboleth

- The Origin
  - Internet2 in the US launched the open source project in 2000
- The name
  - The term ‘shibboleth’ was used to identify members of a group (different dialects had different pronunciations)
- The standard
  - Based on Security Assertion Markup Language (SAML)
- The Consortium
  - The new home for Shibboleth development
  - Collect financial contributions from deployers, worldwide
- There are other products, too (e.g. SimpleSAMLphp, ADFS)
- The Shibboleth software is widely used in the research and education environment
  - https://shibboleth.net/

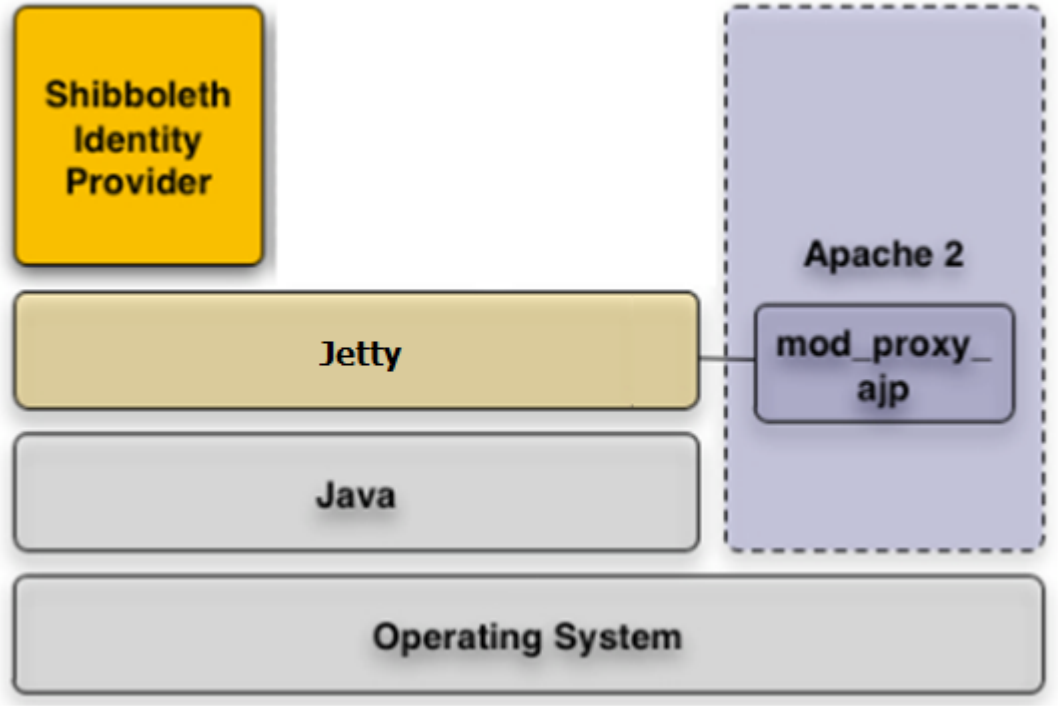
Shibboleth Components

- Most people think of it as a set of software components:
  - OpenSAML C++ and Java libraries
  - Shibboleth Identity Provider (IdP)
  - Shibboleth Service Provider (SP)
  - Shibboleth Discovery Service (DS)
  - Shibboleth Metadata Aggregator (MA)
- Together, these components make up a federated identity management (FIM) platform
- The Shibboleth software components are an implementation of the SAML protocols and bindings



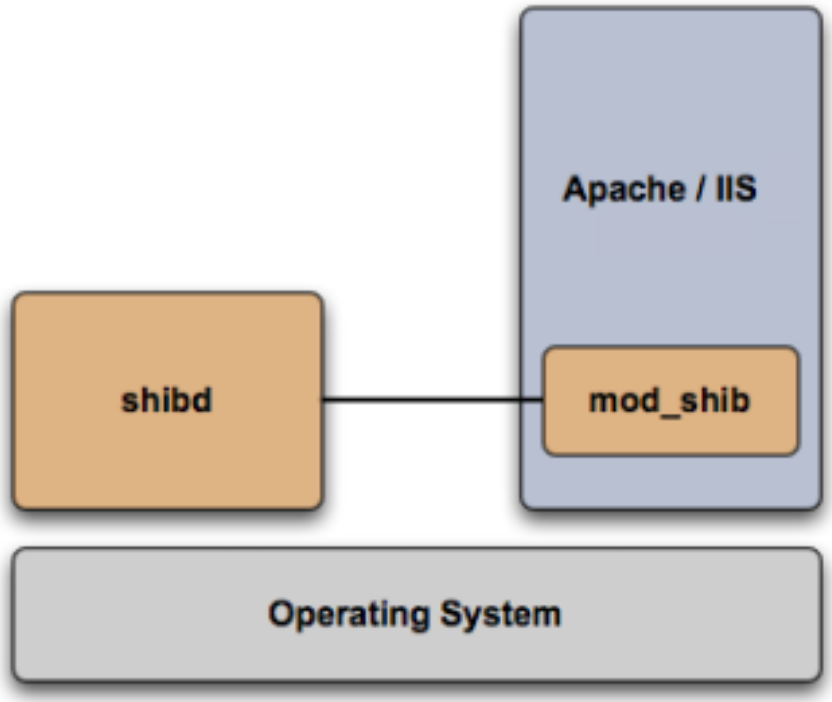
Shibboleth Components: Identity Provider (IdP)

- A Java Servlet web application
- Authenticates users and provides information about users (attributes)
- Connects to existing authentication and user data systems
- Provides information about how a user has been authenticated
- Provides user identity information from the data source



Shibboleth Components: Service Provider (SP)

- mod_shib: C++ web server (Apache/IIS) module
- Shibd: C++ daemon keeps state when web server processes die:
  - Typically initiates the request for authentication and attributes
  - Processes incoming authentication and attribute information (SAML assertion from IdP)
  - Optionally evaluates content access control rules
  - Passes user information (attributes) to web application



Shibboleth Components: Discovery Service (DS)

- Lets the user choose his/her home organisation
- Tells the Service Provider which Identity Provider to use for authentication and attribute retrieval
- Can be integrated into the web resource or used as a separate central service
- Also known as Where Are You From (WAYF) service

Main Goals of Hands-On Session

- Install and configure a Shibboleth Service Provider 2
- Know how and where to configure things
- Learn how to protect web content
- Understand how attributes can be used in web applications

Essential OS Commands for Linux

DOS Command	Linux Command
`dir`	`ls -l`
`cd <directory>`	`cd <directory>`
`mkdir` or `md <directory>`	`mkdir <directory>`
`rmdir` or `rd <directory>`	`rmdir <directory>`
`chdir`	`pwd`
`del` or `erase <file>`	`rm <file>`
`copy` and `xcopy <file>`	`cp` and `cp -R <file>`
`find` or `findstr <file>`	`grep <string> <file>`
`comp <file1> <file2>`	`diff <file1> <file2>`
`edit <file>`	`nano` or `vim` or `emacs <file>`
`ping <host>`	`ping <host>`
`reboot`	`reboot`

Tips and Tricks for Hands-On Session

- The password usually is "password"
- Lines starting with `$` are commands to be executed:
  - Replace `#` with a number (your participation number during the training)
- Command should be executed as root user:
  - Happens automatically if terminal is opened or if text editor is used
- Character `\` is line break symbol, which allows a line to break when typed

More Tips and Tricks for Hands-On Session

- Restart the Shibboleth daemon shibd after every change:
  - shibd automatically reloads config but only restarts "reveal" errors
  - Alternatively, look at the log file for errors
- Delete session cookies after changes (or restart browser):
  - Should not be necessary but is safer for testing
- SSH access to connect to your VM (only with VirtualBox):

  `$ ssh -p 2222 sp-admin@127.0.0.1`

  The password is 'password', user is in sudoers list

  Useful for `$ tail -f /var/log/shibboleth/shibd.log`

VM Operating System Environment

- Ubuntu 14.04 LTS, Virtual Box/VMWare VDK image
- User: “sp-admin" / Password: “password" (in sudoers list)
- Apache 2 on ports 80 (http) and 443 (https)
- Self-signed SSL web server certificate
- Hostname: `sp#.example.org`

VM Setup

1. Open Training image in Virtual Box / VMWare
2. Start the virtual machine (VM)
3. Firefox will open automatically page https://aarc-project.eu/
4. Open terminal 
5. Enter "password" to become root user
6. Execute: `$ setupVM`
7. Enter your training participation number:

   VM will then reboot automatically after a few seconds

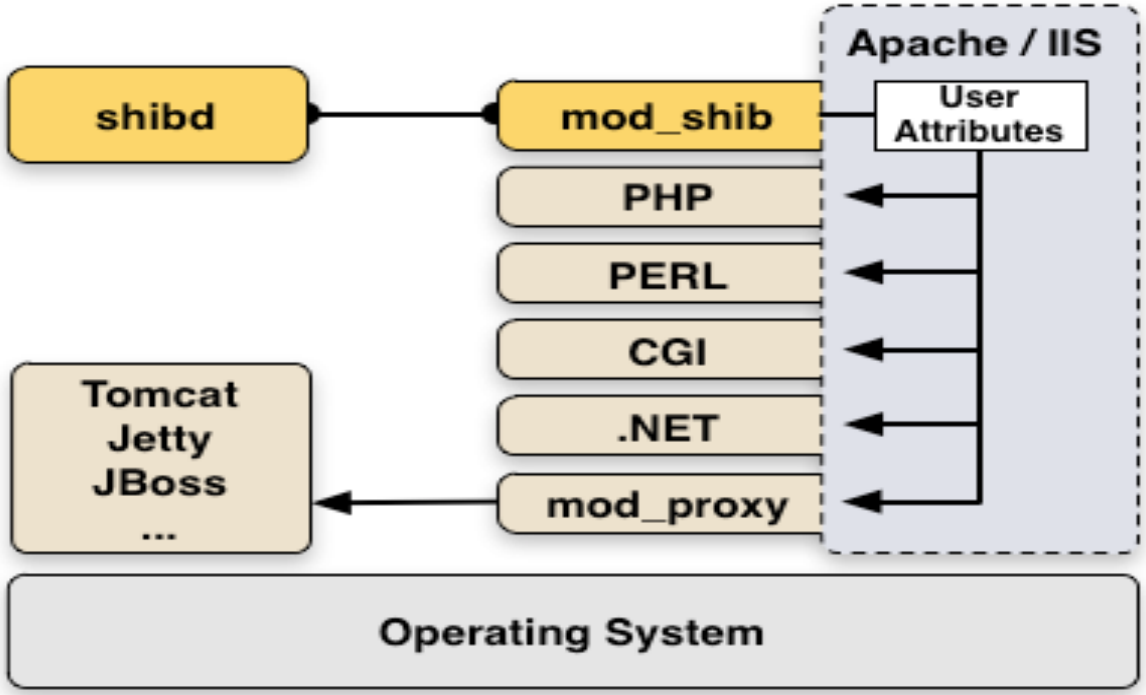
SP Overview and Installation

Goals:

1. Terminology and SP Overview
2. Installation
3. Configuration
4. Quick Sanity Check

Shibboleth SP: Daemon and mod_shib

- Runs on: Linux, Solaris, Windows, Mac OS X, FreeBSD, …
- Protects web applications
- shibd processes attributes
- Can authorise users with
  - Apache directives
  - Shibboleth XML Access rule
- Provides attributes to applications



Terminology

- **Service Provider (SP)**
  Consumes SAML assertions, protects web applications
- **Identity Provider (IdP)**
  Asserts digital identities using SAML
- **Discovery Service (DS/WAYF)**
  Lets user choose Identity Provider/home organisation
- **shibd** (Shibboleth daemon)
  SP service/daemon for maintaining state
- **Session**
  Security context and cached data for a logged-in user
- **Session Initiator**
  Part of SP that controls how SSO requests are started

Service Provider Installation 1/3

1. Open terminal 
2. Define used repository
   - The Shibboleth project only provides official binary packages for RPM-based Linux distributions, SWITCH Ubuntu LTS release is used in this training (Use of the repository is permitted but it is provided AS IS, without any support)

   1. Download repository key
      - `$ sudo curl -k -O http://pkg.switch.ch/switchaai/SWITCHaai-swdistrib.asc`

   2. Verify repository key
      - `$ gpg --with-fingerprint SWITCHaai-swdistrib.asc`
      - verify that the fingerprint of the repository signing key is:
        294E 37D1 5415 6E00 FB96 D7AA 26C3 C469 15B7 6742

   3. Add repository key
      - `$ sudo apt-key add SWITCHaai-swdistrib.asc`

Service Provider Installation 2/3

4. Add repository for Ubuntu 14.04 LTS (trusty)
   - Create a new source file /etc/apt/sources.list.d/SWITCHaai-swdistrib.list
   - `$ echo 'deb http://pkg.switch.ch/switchaai/ubuntu trusty main' | sudo tee /etc/apt/sources.list.d/SWITCHaai-swdistrib.list > /dev/null`
5. Refresh repository
   - `$ sudo apt-get update`

3. Install Shibboleth Service Provider
   - `$ sudo apt-get install shibboleth`
   - answer ”Y” to continue when asked

4. Quick test
   - quick test shows whether the Service Provider was installed properly
   - $ sudo shibd –t
     - Verify that the last line of the output is:

```
overall configuration is loadable, check console for non-fatal problems
```


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

Service Provider Installation 3/3

- Test the Apache configuration
  - `$ sudo apache2ctl configtest`
- Verify that the output is: `Syntax OK`

- Mod_shib test
  - Open Firefox in VM, access the URL: `https://sp#.example.org/Shibboleth.sso/Session`
  - The web server (or Shibboleth module respectively) should return a page that says:

    `A valid session was not found.`

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

Important Service Provider Files and Directories

- `/usr/sbin/shibd`
  - Shibboleth Daemon binary
- `/usr/lib/shibboleth/*.so`
  - Shibboleth Libraries/Modules/Extensions:
- `/etc/shibboleth/`
  - Master(`shibboleth2.xml`) and supporting configuration files
  - Locally maintained metadata files
  - HTML templates (to customise the look & feel of service)
  - Logging configuration files (*.logger)
  - Credentials (certificates and private keys)
- `/var/run/shibboleth/` and `/var/cache/shibboleth/`
  - UNIX socket
  - remote metadata backups
- `/var/log/shibboleth/`
  - shibd.log and transaction.log files
- `/var/log/apache2/` or `/var/log/shibboleth/apache2/`
  - `native.log` (is written by mod_shib web server module)

EntityID of an Service Provider

- Every SP needs a unique identifier: The **entityID**

- Where is entityID used?
  - In transmitted messages, local configuration, metadata
  - IdP log files, configuration, filtering policies

- EntityID should be: Unique, locally scoped, representative and unchanging

Generate X.509 Key/Certificate for SP

- Run script to generate certificate and private key:
  - `$ cd /usr/sbin/`
  - `$ ./shib-keygen -h sp#.example.org -y 3 -e https://sp#.example.org/shibboleth -o /etc/shibboleth/`

- Results in `sp-cert.pem` and `sp-key.pem` in directory /etc/shibboleth/
  - Have a look at the PEM encoded certificate:
    `$ less /etc/shibboleth/sp-cert.pem`
  - To see the detailed content of certificate:
    `$ openssl x509 -text -in /etc/shibboleth/sp-cert.pem`

Configure Service Provider

- Open the main configuration file /etc/shibboleth/shibboleth2.xml
  - `$ vi /etc/shibboleth/shibboleth2.xml`
    - Line 23 set the entityID value according to your SP:
      `https://sp#.example.org/shibboleth`
    - Line 44 set the entityID value of the used Identity Provider (AAI test IdP):
      `https://aai-demo-idp.switch.ch/idp/shibboleth`
    - Line 86 add MetadataProvider section according to used IdP:
      `<MetadataProvider type="XML" file="/opt/shibboleth-idp/metadata/metadata.aaitest.xml"/>`
- Check Shibboleth SP configuration
  - `$ shibd -t`
    - The last line of the outcome should be:

```
overall configuration is loadable, check console for non-fatal problems
```

- Restart Shibboleth SP
  - `$ /etc/init.d/shibd restart`
  - Check the `shibd.log` located in `/var/log/shibboleth/`

Sanity Checks

- Start processes:
  - `$ /etc/init.d/shibd restart`
  - `$ /etc/init.d/apache2 restart`
- Check shibd status (XML should be returned on success):

```
$ curl -k --interface lo \
  https://sp#.example.org/Shibboleth.sso/Status
```

- Access session handler from your browser:

```
https://sp#.example.org/Shibboleth.sso/Session
```

- See how a Shibboleth error looks like (you get an exception):

```
https://sp#.example.org/Shibboleth.sso/Foobar
```

Bootstrapping the SP

Goals:

1. First attempt to login on Service Provider
2. Learn about Metadata
3. Add Service Provider into AAI Test Federation

Configuration Files

- Service Provider is now installed and configured
  - Let's see if authentication already works

- Open Firefox in VM:
  `https://sp#.example.org/Shibboleth.sso/Login`
  - `/Shibboleth.sso/Login` is the default login initiator
    - It can be used to start the login process
    - You will see an error message:
      - Used AAI Test Identity Provider don't "know" your Service Provider yet because it don't have the metadata about your SP

(Federation) Metadata

- SAML Metadata is an XML document
- Typically is provided by a federation operator
- Contains descriptions of all SPs and IdPs:
  - **entityID:** The unique identifier of the entity
  - **Supported protocols:** E.g. SAML1, SAML2
  - **X.509 certificates:** Contain the public key of a key pair
  - **Endpoint URLs:** What URLs to query or send messages to
  - **Descriptive** information: e.g. display name, description, logos
  - **Contact information:** e.g. for support

- Now also have a look at your SP's metadata by opening:
  `https://sp#.example.org/Shibboleth.sso/Metadata`
  SP can generate (technical) SAML metadata about itself

Login to AAI Resource Registry

- By using Firefox navigate to `https://rr.aai.switch.ch`

1. Login to AAI Resource Registry:
   - Click on 

   - Select the "AAI Demo Home Organisation" IdP from list
   - To authenticate use Username: `sp-training-admin`, password: `password`

2. Click on the "Resources" tab

3. Then click on "Add a Resource Description"

Note: The description of your training SP will be deleted at some point after the training

Test SP Resource Description 1/2

1. Open in Firefox: `https://sp#.example.org/Shibboleth.sso/Metadata`
   - This should open the XML file in the gedit text editor
2. Copy all XML and paste it into the text area on the Resource Registry above the button "Run Shibboleth 2.x wizard"
3. Then hit the button "Run Shibboleth 2.x wizard“
   - "Descriptive Information", "Service Locations" and "Certificates" should now be green (=completed)
4. Click on "Basic Resource Information":
   - Choose "aai-demo-idp.switch.ch" as Home Organisation.
   - Add a name and description for your Service Provider
     - e.g. "Demo SP #", "SP used for improving my Shib knowledge"
   - Finally, click on "Save and Continue"
5. Click on "Contacts“
   - Complete the fields with your email address and contact info.
   - Finally, click on "Save and Continue"

Test SP Resource Description 2/2

6. Click on "Requested Attributes“
   - Add as required attributes: Targeted ID/Persistent ID, Surname, Given Name, E-Mail, Affiliation, Entitlement and PrincipalName
   - Finally, click on "Save and Continue"
7. Click on "Intended Audience”:
   - Click on "Set all to included".
   - Finally, click on "Save and Continue"
   - You now have completed the Resource Registry
8. Add a comment that you are registering this Resource Description in the context of the SP training
9. Click on "Submit for Approval"
   - You should then see the Resource Registration Authority (RRA) administrators, who have to approve your Resource Description
   - The RRA admins might ask you for the certificate fingerprint to prove that you generated the certificate
   - To get the certificate's SHA1 fingerprint, run:

```
Openssl x509 -fingerprint -in /etc/shibboleth/sp-cert.pem
```

Demo Users on AAI Test Identity Provider

- Username: `g.utente` Password: `password`
  - Givenname surname: `Giovanni Utente`
  - Affiliation: `faculty;member`
  - Entitlements: `http://example.org/res/99999, http://publisher-xy.com/e-journals`

- Username: `p.etudiant` Password: `password`
  - Givenname surname: `Pière Edudiant`
  - Affiliation: `student;member`
  - Entitlements: `urn:mace:dir:entitlement:common-lib-terms, http://www.example.org/aai/agreement-2011`

- Username: `h.mitarbeiter` Password: `password`
  - Givenname surname: `Hans Mitarbeiter`
  - Affiliation: `staff;student;member`
  - Entitlements: `urn:mace:dir:entitlement:common-lib-terms, http://www.example.org/vip`

Test Access

- SP's metadata is included into AAI test federation's metadata
  - During the training event, metadata propagation is max. 5 minutes

1. In Firefox, open the URL: `https://sp#.example.org/Shibboleth.sso/Login`
2. Select the "AAI Demo Home Organisation" on the WAYF
   - If you instead get an error, wait a few minutes more
3. Use “`g.utente`" and "`password`" as login name and password

Service Provider Deployment Complete

- When you are back on the Shibboleth Session handler (`/Shibboleth.sso/Session`) and see something like...

```
Miscellaneous
Session Expiration (barring inactivity): 479 minute(s)
Client Address: 127.0.0.1
SSO Protocol: urn:oasis:names:tc:SAML:2.0:protocol
Identity Provider: https://aai-demo-idp.switch.ch/idp/shibboleth
Authentication Time: 2015-12-22T10:55:58.914Z
Authentication Context Class: urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Authentication Context Decl: (none)

Attributes
entitlement: 2 value(s)
persistent-id: 1 value(s)
unscoped-affiliation: 1 value(s)
```

... then you have successfully deployed your Service Provider

Basic Configuration

Goals:

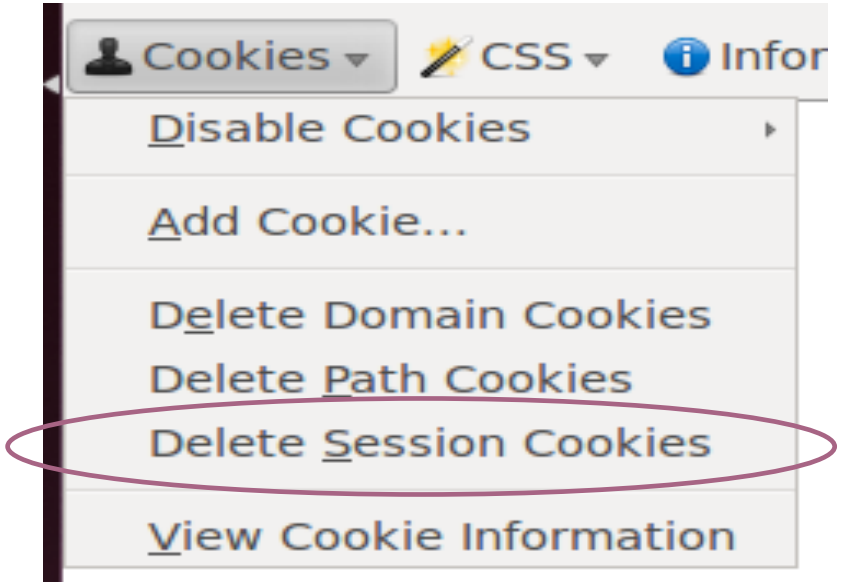
1. Local logout
2. Understand purpose and structure of SP configuration files
3. Increase log level to DEBUG
4. Make a few simple configuration changes

Logging Out

- To logout locally from the SP and kill your session:
  `https://sp#.example.org/Shibboleth.sso/Logout`
  But this won't delete your session on the IdP!

- **Close the browser and restart it!**
  Still the easiest and safest method for most web browsers

- Or delete all your session cookies
  For testing and development purposes use the "Firefox Web Developer" extension (installed on VM)



Configuration Files in /etc/shibboleth

- **`shibboleth2.xml` – main configuration file**
- `attribute-map.xml` – attribute handling
- `attribute-policy.xml` – attribute filtering settings
- `*.logger` – logging configuration
- `*Error.html` –HTML templates for error messages
- `localLogout.html` – SP-only logout template
- `globalLogout.html` – single logout template

Recommendation:
Adapt *.html files for production configuration to match the look and feel of the protected application improves user experience.

Shibboleth2.xml Structure

Since Shibboleth 2.4: Simplified configuration but old format still accepted

`<SPConfig>`	Document root element

Outer elements of the shibboleth.xml configuration file:

`<OutOfProcess> / <InProcess>`	(Optional) Log settings, extensions
`<UnixListener> / <TCPListener>`	(Optional) Communication shibd/mod_shib
`<StorageService>`	(Optional) Where session information is stored
`<SessionCache>`	(Optional) Session timeouts and cleanup intervals
`<ReplayCache>`	(Optional) Where replay cache is stored
`<ArtifactMap>`	(Optional) Timeout of artifact messages
`<RequestMapper>`	(Optional) Session initiation and access control
`<ApplicationDefaults>`	**Contains the most important settings of SP**
`<SecurityPolicyProvider>`	Define various security options
`<ProtocolProviders>`	Defines supported protocols (SAML, ADFS, ...)

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

ApplicationDefaults Structure

You are most likely to modify <ApplicationDefaults>:

<Sessions>	Defines handlers and how sessions are initiated and managed. Contains <SSO>, <Logout>, <Handler>
<Errors>	Used to display error messages. E.g. logo, email and CSS
<RelyingParty>	(optional) To modify settings for certain IdPs/federations
<MetadataProvider>	Defines the metadata to be used by the SP
<AttributeExtractor>	Attribute map file to use
<AttributeResolver>	Attribute resolver file to use
<AttributeFilter>	Attribute filter file to use
<CredentialResolver>	Defines certificate and private key to be use
<ApplicationOverride>	**(Optional) Can override any of the above for certain applications**

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

File Editing Commands for Terminal Editor

Editor	nano	vim
Open file	`$ nano <file>`	`$ vim <file>`
Save file	<ctrl>-o	<esc>, :w
Save and exit	<ctrl>-x	<esc>, :wq
Search string	<ctrl>-w, **string**	<esc>, /**string**
Go to line number	<ctrl>--, **number**	<esc>, **number**, <shift>-G

gedit is the recommended text editor. It is started as root user
Its icon is in the launch bar on the left side of the desktop

Debugging SP Problems on Linux

1. Make sure the edited XML config file is valid and correct XML with:
   ```
   $ xmlwf /etc/shibboleth/shibboleth2.xml
   $ sudo shibd -t   or
   $ sudo shibd -tc /etc/shibboleth/shibboleth2.xml
   ```
2. Stop Shibboleth daemon with:
   ```
   $ /etc/init.d/shibd stop
   ```
3. Increase log verbosity of shibd by seting log level to DEBUG in:
   ```
   /etc/shibboleth/shibd.logger
   ```
4. Have a look at log file and search ERROR or CRIT messages in:
   ```
   $ tail -f /var/log/shibboleth/shibd.log
   ```
5. Start Shibboleth daemon again with:
   ```
   $ /etc/init.d/shibd start
   ```
6. If you fixed an error, also restart Apache with:
   ```
   $ /etc/init.d/apache2 restart
   ```

Don't forget to set log level back to `INFO` for a production service!

Logging

- Your number one friend in case of problems
- `shibd.log` and `transaction.log` written by shibd, `native.log` written by mod_shib
- `*.logger` files contain predefined settings for output locations and a default logging level (`INFO`) along with useful categories to raise to `DEBUG`

Increase Log Level to DEBUG

Raise categories:

- `$ vim /etc/shibboleth/shibd.logger`
- `Line 2:`
  `log4j.rootCategory=`**`DEBUG,`** `shibd_log, warn_log`

- Starting in `Line 16`, take into use:

```
# tracing of SAML messages and security policies
log4j.category.OpenSAML.MessageDecoder=DEBUG
log4j.category.OpenSAML.MessageEncoder=DEBUG
log4j.category.OpenSAML.SecurityPolicyRule=DEBUG
```

Make Session Handler Show Values

- For debugging purposes, it helps seeing the attribute values on `/Shibboleth.sso/Session`

- Open the `/etc/shibboleth/shibboleth2.xml`

```
Line 59:
<!-- Session diagnostic service. -->
<Handler type="Session" Location="/Session" showAttributeValues="true"/>
```

Apply Configuration Changes

- To follow the shibd log file in real time and examine what happens during a login use tail:
  - `$ tail -f /var/log/shibboleth/shibd.log`

- `shibd` reloads configuration when `shibboleth2.xml` is changed, but generally it is still better to restart shibd:
  - `$ /etc/init.d/shibd restart`

  - Shibboleth detects invalid configurations if it reloads them automatically
  - In this case it will continue to use the last valid configuration it still has in memory
  - This behavior hides errors that will only be discovered at the next restart!

Check Changes

- Login again with:
  `https://sp#.example.org/Shibboleth.sso/Login`
- In the log you should see the decoded XML assertion received by SP:

```
DEBUG Shibboleth.SSO.SAML2 [1]: decrypted Assertion: <saml2:Assertion
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
ID="_efc943c04c742ae96d15e19e95afba68" IssueInstant="2015-12-10T14:59:29.841Z" Versi
on="2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">[...]
```

- The Session Handler should now also display the attribute values:
  `https://sp#.example.org/Shibboleth.sso/Session`

```
Attributes
entitlement: http://example.org/res/99999;http://publisher-xy.com/e-journals
persistent-id: https://aai-demo-idp.switch.ch/idp/shibboleth!https://sp77.example.or
unscoped-affiliation: faculty
```

SP Metadata Features

- Metadata describes the other components (IdPs) that the Service Provider can communicate with

- **Four primary methods built-in:**
  - **Local metadata file (you download/edit it manually)**
  - **Downloaded remotely from URL (periodic refresh, local backup)**
  - Dynamic resolution of entityID (=URL), hardly used
  - "Null" source that disables security ("OpenID" model), hardly used

- Security comes from metadata filtering, either by you or the SP:
  - Signature verification
  - Expiration dates
  - White and blacklists

Add Metadata Signature Verification

- Update MetadataProvider section in line 86 in `/etc/shibboleth/shibboleth2.xml`:

```
<MetadataProvider type="XML" file="/opt/shibboleth-idp/metadata/metadata.aaitest.xml">
   <MetadataFilter type="Signature">
      <TrustEngine type="StaticPKIX"
         certificate="/opt/shibboleth-idp/credentials/SWITCHaaiRootCA.crt.pem"
         verifyDepth="2" checkRevocation="fullChain" policyMappingInhibit="true"
         anyPolicyInhibit="true">
         <TrustedName>SWITCHaai Metadata Signer</TrustedName>
      </TrustEngine>
   </MetadataFilter>
</MetadataProvider>
```

- Restart Shibboleth and you should see lines in `/var/log/shibd.log` – file about signature verification

  2016-01-19 11:50:37 INFO OpenSAML.Metadata : applying metadata filter (Signature)

  2016-01-19 11:50:37 DEBUG XMLTooling.TrustEngine.PKIX : validating signature using certificate from within the signature

  2016-01-19 11:50:38 DEBUG XMLTooling.PathValidator.PKIX : successfully validated certificate chain

Attribute Handling

Goals:

1. Learn how attributes are mapped and filtered
2. See how attributes can be used as identifiers
3. Add an attribute mapping and filtering rule

Attribute Mappings

- SAML attributes from any source are "extracted" using the configuration rules in attribute map file in:
  `/etc/shibboleth/attribute-map.xml`

- Each element is a rule for decoding a SAML attribute and assigning it a local `id`, which becomes its mapped variable name

- Attributes can have one or more `id` and multiple attributes can be mapped to the same `id`

- The `id` is also used as header name in the webserver for this attribute

Add Mail Attribute

- Open attribute-map.xml:
  - `$ vim /etc/shibboleth/attribute-map.xml`

  Map mail attribute `Line 11:`

  ```
  <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
  ```

- Restart service
  - `$ /etc/init.d/shibd restart`

- Login and check your session, mail should be now mapped:
  - `https://sp#.example.org/Shibboleth.sso/Session`

```
Attributes
entitlement: http://example.org/res/99999;http://publisher-xy.com/e-journals
mail: g.utente@example.org
persistent-id: https://aai-demo-idp.switch.ch/idp/shibboleth!https://sp77.example.org/shi
unscoped-affiliation: faculty
```

Add Name Attributes

- Open attribute-map.xml:
  - `$ vim /etc/shibboleth/attribute-map.xml`

  Add (or uncomment) name attributes before mail attribute in `line 11`:

```
<Attribute name="urn:oid:2.5.4.4" id="sn"/>
<Attribute name="urn:oid:2.5.4.42" id="givenName"/>
```

- Restart service
  - `$ /etc/init.d/shibd restart`

- Login and check your session, name attributes should be now mapped:
  - `https://sp#.example.org/Shibboleth.sso/Session`

```
Attributes
entitlement: http://example.org/re
givenName: Giovanni
mail: g.utente@example.org
persistent-id: https://aai-demo-ic
sn: Utente
unscoped-affiliation: faculty
```

Attribute Filtering

- Answers the "who can say what" question on behalf of an application
- Service Provider can make sure that only allowed attributes and values are made available to an application
- Some examples:
  - Constraining the possible values or value ranges of an attribute (e.g. eduPersonAffiliation, telephoneNumber, ...)
  - Limiting the scopes/domains an IdP can speak for (e.g. university x cannot assert faculty@university-z.edu)
  - Limiting custom attributes to particular sources

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

Default Filter Policy

- As default, **attributes are filtered out unless there is a rule!**
  - `/etc/shibboleth/attribute-policy.xml`
- Shared rule for legal affiliation values
- Shared rule for scoped attributes
- Generic policy applying those rules and letting all other attributes through
- Check `/var/log/shibboleth/shibd.log` for signs of filtering in case of problems with attributes not being available.
  You would find something like "`no values left, removing attribute (#attribute name#)`"

Add Rule for Surname Attribute

- Add a rule to limit acceptance of "surname" to a single IdP:
  - $ vim /etc/shibboleth/attribute-policy.xml

```
Line 61:
<afp:AttributeRule attributeID="sn">
  <afp:PermitValueRule xsi:type="AttributeIssuerString"    value="https://
  non.existing.idp.example.org/idp/shibboleth"/>
</afp:AttributeRule>
```

- Restart Shibboleth SP
  - $ /etc/init.d/shibd restart

- Login and check your session
  - https://sp#.example.org/Shibboleth.sso/Session
  - surname is now filtered out and cannot be seen but other attributes aren't. Because a specific rule exists for surname, the catch-all rule at the bottom of the file does not apply anymore!

Remove Surname Specific Rule

- Remove earlier created rule for `surname(sn)`:

```
$ vim /etc/shibboleth/attribute-policy.xml
```

- Restart Shibboleth SP
  - `$ /etc/init.d/shibd restart`

- Login and check your session
  - `https://sp#.example.org/Shibboleth.sso/Session`
  - Now you should see the `surname` attribute again

Session Initiation

Goals:

1. Learn how to initiate a Shibboleth session
2. Understand their advantages/disadvantages
3. Know where to require a session, what to protect

Content Protection and Session initiation

- Before access control (will be covered later on) can occur, a Shibboleth session must be initiated
- Session Initiation and content protection go hand in hand
- Requiring a session means the user has to authenticate
- Only authenticated users can access protected content

Content Protection Settings

Protect hosts, directories, files or queries

- **Apache**
  .htaccess (dynamic) or httpd.conf (static)

- **Apache / IIS / other**
  <RequestMap> in shibboleth2.xml
  Requires Shibboleth to know exact hostname
  Very powerful and flexible thanks to boolean/regex operations

- Try accessing `https://sp#.example.org/secure/`
  You should get access because the directory is not protected (yet)
  `/secure/` used to be protected by default in older Shibboleth distributions

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

Content Protection with .htaccess File

Let’s protect the directory by requiring a Shibboleth session:

- `$ vim /var/www/secure/.htaccess`

```
AuthType shibboleth
require shibboleth
ShibRequestSetting requireSession true
```

- Rules could also be in static Apache configuration file under Apache directory `/etc/apache2/`

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

Session Initiation and Content Settings

- **forceAuthn** `(ShibRequestSetting forceAuthn true)`
  - Disable Single-Sign on and force a re-authentication
- **isPassive** `(ShibRequestSetting isPassive true)`
  - Check whether a user has an SSO session and if he has, automatically create a session on SP without any user interaction
- Use a specific IdP to use for authentication
- Requesting types of authentication
  - e.g enforce X.509 user certificate authentication
- Custom error handling pages to use
- Redirection-based error handling
  - In case of an error, redirect user to custom error web page with error message/type as GET arguments

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

Test Content Protection Rule

- Clear the session and then access the protected URL again:
  - `https://sp#.example.org/secure`
- Authentication is enforced and access should be granted
- Currently, all authenticated users get access
- Content protection to limit access only to specific users will be covered later

Content Protection with RequestMap

- `mod_shib` provides request URL to `shibd` to process it
  Therefore, `shibd` can enforce access control as well

- First ensure that requests for `/other-secure/` are handled by `shibd` without setting any specific session requirements:

- `$ vim /var/www/other-secure/.htaccess`

```
AuthType shibboleth
require shibboleth
```


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

How to Add a RequestMap

- Open the Shibboleth configuration:

```
$ vim /etc/shibboleth/shibboleth2.xml
```

Before `<ApplicationDefaults>` insert a `<RequestMap>`:

Line 7:

```
<RequestMapper type="Native">
  <RequestMap applicationId="default">
    <Host name="sp.example.org">
      <Path name="other-secure" authType="shibboleth" requireSession="true"/>
    </Host>
  </RequestMap>
</RequestMapper>
```

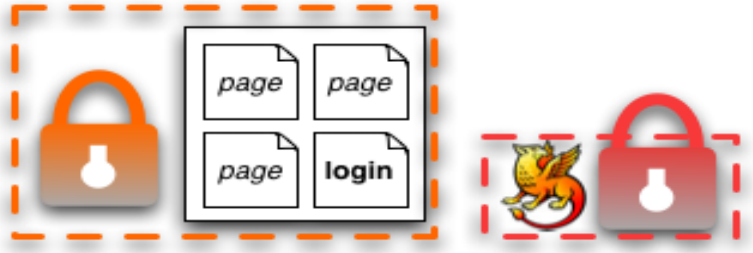
- Clearing session and then accessing `/other-secure/` now, one also is forced to authenticate

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

Where to Require a Shibboleth Session

- **Whole application with "required" Shibboleth session**
  - Easiest way to protect a set of documents
  - No other authentication methods possible
  - Needs special config to preserve HTTP POST requests

- **Whole application with "lazy" Shibboleth session**
  - Also allows for other authentication methods
  - Authorisation can only be done in application



- **Only page that sets up application session**
  - Well-suited for dual login
  - Application can control session time-out
  - **Generally the most popular solution (many Shibboleth-enabled applications use this)**

Protect a Simple Page or Web Application

- Access `https://sp#.example.org/cgi-bin/attribute-viewer`
  Simple CGI script as a sample application that show attributes

- Lets protect that script with Shibboleth by requiring a session:

```
$ vim /usr/lib/cgi-bin/.htaccess

AuthType shibboleth
ShibRequestSetting requireSession true
require shibboleth
```

  - This will require a session for all requests to `/cgi-bin/` and make attributes available to application in environment.

- Try again to access script with a browser:
  - Script should enforce authentication and show attributes

Make Script "see" Shibboleth Session

- What if we wanted to grant access also to non-authenticated users, but use attributes if somebody is authenticated?
- Use Shibboleth (lazy) session:

```
$ vim /usr/lib/cgi-bin/.htaccess

AuthType shibboleth
require shibboleth
```

  - This will not require a session but make attributes available to application in environment if somebody has a session

- Try again with a browser:

```
https://sp#example.org/cgi-bin/attribute-viewer
```

  - Unauthenticated access still possible. No attributes are shown yet

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

REMOTE_USER

- Special single-valued variable that all web applications should support for container-managed authentication of a unique user
- Any attribute, once extracted/mapped, can be copied to REMOTE_USER
- Multiple attributes can be examined in order of preference, but only the first value will be used

Changing REMOTE_USER

- In case your application needs to have a remote user for authentication, you just could make shibboleth put an attribute (e.g. "mail") as REMOTE_USER:

```
/etc/shibboleth/shibboleth2.xml

<ApplicationDefaults>
  REMOTE_USER="mail eppn persistent-id targeted-id"
```

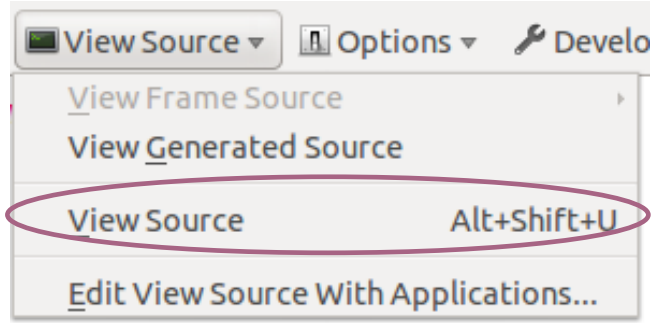
- If mail attribute is available, it will be put into REMOTE_USER

- Attribute `mail` has precedence over `persistent-id` in this case
- This allows very easy "shibbolisation" of some web applications

How To Initiate a (Lazy) Session

- Close your browser, and access the attribute-viewer again:
  - `https://sp#.example.org/cgi-bin/attribute-viewer`
- Then click the login button and login using Test IdP:
  - You should be sent to IdP and attribute-viewer should display attributes after successful authentication
- Have a look at the HTML source and what it does:
  - `https://sp#.example.org/cgi-bin/attribute-viewer`
- Script initiates Shibboleth session by sending user to

  `/Shibboleth.sso/Login``?target=/cgi-bin/attribute-viewer`
  `&entityID= https://aai-demo-idp.switch.ch/idp/shibboleth`



Try to Initiate a Session Yourself

- Try to construct a Session Initiation URL yourself by using these parameters to see the result: e.g. try supplying the IdP:

```
https://sp#.example.org/Shibboleth.sso/Login?
  target=https://sp#.example.org/cgi-bin/attribute-viewer&
  entityID=https://https://aai-demo-idp.switch.ch/idp/shibboleth
```

- This way, a session using a specific IdP can be initiated directly with a link, e.g. on a portal web page
- This allows creating direct "login links" (to skip the Discovery Service)
- It also allows overriding certain content settings

Session Creation Parameters

- Key parameters:
  - target (defaults to homeURL or "/")
  - entityID (specific IdP to use or WAYF/DS if not present)

- Most parameters can be set at three places. In order of precedence:
  - In query string parameter of a URL to handler
  - a content setting (.htaccess or RequestMap)
  - <SessionInitiator> element

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

Lazy Sessions Summary

- Won’t enforce a Shibboleth session but use it if it is available:
  - If valid **session exists**:
    - Process it as usual (put attributes in server environment, etc.)
  - If a **session does NOT exist** or is invalid:
    - Ignore it and pass on control to application

- Three common cases:
  - Public and private access to the same resources
  - Separation of application and SP session
  - Dual login (use Shibboleth and some other authentication method)

Using Lazy Sessions

- In place of an API to "doLogin", the SP uses redirects
  - `https://sp#.example.org/Shibboleth.sso/Login`

- When your application wants a login to happen, redirect the browser to a SessionInitiator (`/Shibboleth.sso/Login` by convention) with any parameters you want to supply

Access Control

Goals:

1. Create some simple access control rules
2. Get an overview about the three ways to authorise users
3. Understand their advantages/disadvantages

Access Control

- Integrated in Service Provider via an AccessControl API built into the request processing flow
- Two implementations are provided by the SP:
  - .htaccess "require" rule processing
  - XML-based policy syntax attached to content via RequestMap
- Third option: Integrate access control into web application

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

Access Control Mechanisms

	1.a httpd.conf	1.b .htaccess	2. XML AccessControl *	3. Application Access Control
⊕	▪ Easy to configure ▪ Can also protect locations or virtual files ▪ URL regex	▪ Dynamic ▪ Easy to configure	▪ Platform independent ▪ Powerful boolean rules ▪ URL Regex ▪ Dynamic	▪ Very flexible and powerful with arbitrarily complex rules ▪ URL Regex Support
⊖	▪ Only works for Apache ▪ Not dynamic ▪ Very limited rules	▪ Only works for Apache ▪ Only usable with "real" files and directories	▪ XML editing ▪ Configuration error can prevent SP from restarting	▪ You have to implement it yourself ▪ You have to maintain it yourself

* Configured in RequestMap or referenced by an .htaccess file

Apache httpd.conf or .htaccess Files

- Work almost like known Apache "require" rules:
  e.g. `Require shib-attr affiliation staff`
  or `Require shib-attr mail user1@idp1.com user2@idp2.org`

- Special rules:
  - `shibboleth` (no authorisation)
  - `valid-user` (require a session, but NOT identity)
  - `user` (REMOTE_USER as usual)
  - `authnContextClassRef`, `authnContextDeclRef`

- Default is boolean "OR", use `ShibRequireAll` for AND rule

- Regular expressions supported using special syntax:

```
Require shib-attr mail ~ [exp]
Require shib-attr mail ~ ^.*@(it|faculty).example.org$
```

Example .htaccess File

- Require a user to be a staff member:

```
$ vim /var/www/staff-only/.htaccess

AuthType shibboleth
ShibRequestSetting requireSession true
Require shib-attr unscoped-affiliation staff
```

- Access: `https://sp#.example.org/staff-only/`
  with user `h.mitarbeiter` and access should be granted.

- Access: `https://sp#.example.org/staff-only/`
  with user `p.etudiant` and access should be denied

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

XML Access Control

- Can be used for access control independent from web server and operating system
- XML Access control rules can be embedded inside RequestMap or be dynamically loaded from external file
- Boolean operators (AND,OR,NOT) can be used
- .htaccess files can reference to XMLAccessControl files
  Allows outsourcing access control rules to non-root users

XML Access Control Example

- `$ vim /var/www/students-only/.htaccess`

```
AuthType shibboleth
require shibboleth
```

- Require affiliation value “student” for *students-only* -directory:

  `$ vim /etc/shibboleth/shibboleth2.xml`

  Inside `<RequestMapper type="Native">` -element add:

```
<RequestMap applicationId="default">
  <Host name="sp1.example.org">
    <Path name="students-only" authType="shibboleth" requireSession="true">
      <AccessControl>
        <Rule require="unscoped-affiliation">student</Rule>
      </AccessControl>
    </Path>
  </Host>
</RequestMap>
```

- Access `https://sp#.example.org/students-only/` with user `p.etudiant` and access should be granted

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

Application Managed Access Control

- Application can access and use Shibboleth attributes by reading them from the web server environment
- Attributes then can be used for authentication/access control/authorisation

PHP:

```
if ($_SERVER['affiliation'] == 'staff;member')
    { grantAccess() }
```

Perl:

```
if ($ENV{'affiliation'} == 'staff;member')
    { &grantAccess() }
```

Java:

```
if (request.getHeader("affiliation").equals("staff;member") )
   { grantAccess() }
```

Service Provider Handlers

Goals:

1. Understand the idea of a handler
2. Get an overview about the different types of handlers
3. Know how to configure them if necessary

SP Handlers

- **"Virtual" applications inside the SP with API access:**
  - SessionInitiator (requests)
    - Start Shibboleth sesion: `/Shibboleth.sso/Login`
  - AssertionConsumerService (incoming SSO)
    - Receives SAML assertions: `/Shibboleth.sso/SAML/POST`
  - LogoutInitiator (SP signout)
    - Log out from SP: `/Shibboleth.sso/Logout`
  - SingleLogoutService (incoming SLO)
  - ManageNameIDService (advanced SAML)
  - ArtifactResolutionService (advanced SAML)
  - Generic (diagnostics, other useful features)
    - Returns session information: `/Shibboleth.sso/Session`
    - Returns detailed SP status: `/Shibboleth.sso/Status`
    - Returns SP metadata: `/Shibboleth.sso/Metadata`

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

SP Handlers

- The URL of a handler = handlerURL + the location of the handler

  e.g. for a virtual host sp.example.org with handlerURL of "/Shibboleth.sso", a handler with a Location of `/Login` will be: `https://sp#.example.org/Shibboleth.sso/Login`

- Handlers aren't always SSL-only, but usually should be Recommended to set `handlerSSL="true"` in `shibboleth2.xml`

- Metadata basically consists of entityID, keys and handlers

- Handlers are never "protected" by the SP
  But sometimes by IP address (e.g. with `acl="127.0.0.1"`)

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. 653965 (AARC).