



Shibboleth Service Provider

Shibboleth Service provider workshop



- This work is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).



Acknowledgements

- [SAMLConf]
<http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf>
- Shibboleth 2.0 InstallFest Service Provider Material – Ann Arbor, MI
- SP Hands-on Session – SWITCH
- <https://wiki.shibboleth.net/confluence/display/SHIB2/Home>



Program

- Installation
- Bootstrapping SP
- Configuration



Environment

- Parameters
 - \$WORKSH_IDP: the hostname of your IdP
- Shibboleth standard base for this workshop
 - \$HOSTNAME: hostname of your machine (i.e. shibsp01-ws2.lab.iamfederated.org)
 - \$PKI: /etc/ssl
 - \$WEB_SERVER: /etc/apache2
 - \$WEB_DOCROOT: /var/www/shib
 - \$WEB_LOG: /var/log/apache2
 - \$SHIB_HOME: /etc/shibboleth
 - \$SHIB_CONF: /etc/shibboleth
 - \$SHIB_RUN: /var/run/shibboleth
 - \$SHIB_LOG: /var/log/shibboleth

- Tools
 - An absolute must: Syntax friendly editor

```
$ apt-get install vim  
:syntax on
```

- HTTP client
 - links

- Check your time now!



- Always work case sensitive!

- Key/Certificate generation
 - Webserver
 - Signed by TERENA SSL CA
 - Located at \$PKI
 - Shibboleth SP
 - Self-signed
 - Located at \$SHIB_HOME
- Parse certificates

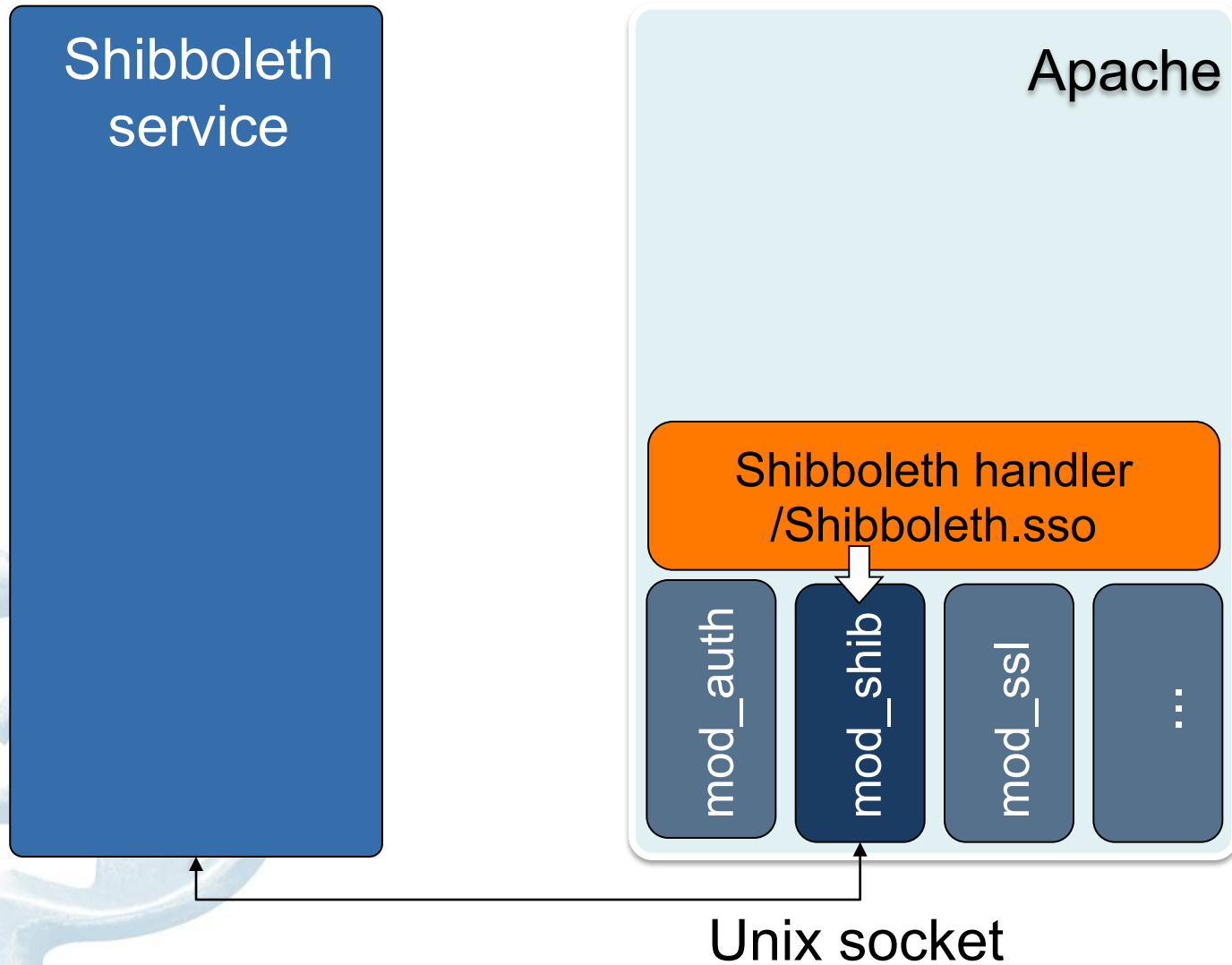
```
openssl x509 -in $cert -issuer -noout
```

SSL certificates

- Use of self-signed certificates in backend
 - No need for commercial certificates 
 - Longer lifetime
 - No truststore to maintain for commercial CAs
 - Revocation (just remove certificate)
 - Trustbase of commercial signed certificates can become quite large
 - Separate certificate for front- and backend



Installation - Overview



Webserver - Installation

- Done for you! 😊

```
$ apt-get install libapache2-mod-php5
```

- DocumentRoot: /var/www (\$WEB_DOCROOT)
- Configuration: /etc/apache2 (\$WEB_SERVER)
- Logs: /var/log/apache2 (\$WEB_LOG)
- Start/Stop/Restart service

```
$ apache2ctl start|stop|restart
```

Webserver - Configuration

- Edit ServerName
- Set UseCanonicalName to On

```
$ vim /etc/apache2/apache2.conf  
    ServerName $HOSTNAME  
    UseCanonicalName On
```

- Otherwise RequestMap could be bypassed!!



Webserver - SSL

```
$ a2enmod ssl  
$ vim /etc/apache2/sites-available/default-ssl
```

```
SSLCertificateFile /etc/ssl/certs/lab.iamfederated.org.crt  
SSLCertificateKeyFile /etc/ssl/lab.iamfederated.org.key  
SSLCertificateChainFile /etc/ssl/lab.iamfederated.org/  
lab.iamfederated.org.chain
```

```
$ a2ensite default-ssl  
$ apache2ctl configtest  
$ /etc/init.d/apache2 restart  
$ openssl s_client -connect localhost:443
```

Webserver - Configuration

- Prepare test application

```
$ a2ensite shibsp  
$ mkdir /var/www/shib/secure  
$ vim /var/www/shib/secure/index.php
```

```
<?php  
header('Location: https://'.$_SERVER['SERVER_NAME'].'/Shibboleth.sso/Session');  
?>
```

Shibboleth SP installation

- Install Shibboleth 2.4.3 (done for you 😊)

```
$ apt-get install libapache2-mod-shib2
```

- Generate self signed certificate

```
$ /usr/sbin/shib-keygen -h $HOSTNAME -e $ENTITYID -y  
$YEARS_VALID
```

- \$ENTITYID = [https://\\$HOSTNAME/shibboleth](https://$HOSTNAME/shibboleth)
- \$YEARS_VALID=3

- Change ownership of key-file

```
$ chown _shibd $SHIB_CONF/sp-key.pem
```

Shibboleth SP installation

- Configuration files provided by deb packages

```
/etc/apache2/mods-available/shib2.load  
/etc/init.d/shibd
```

- **Create** /etc/apache2/mods-available/shib2.conf
- **...or Edit** /etc/apache2/sites-available/shibsp

```
<Location /secure>  
AuthType shibboleth  
require shibboleth  
</Location>
```

```
$ a2enmod shib2  
$ /etc/init.d/shibd restart  
$ /etc/init.d/apache2 restart
```

Sanity checks

- Access Shibboleth handler from your browser
`https://$WORKSH_HOST/Shibboleth.sso`



Session Creation Failure

- Access session handler from your browser
`https://$WORKSH_HOST/Shibboleth.sso/Session`
→ A valid session was not found.

- See how a Shibboleth error looks like
`https://$WORKSH_HOST/Shibboleth.sso/Foo`



Session Creation Failure

Program

- Installation
- Bootstrapping SP
- Configuration



Bootstrapping the SP

Goals:

1. Working SP against a single IdP
2. Enable debugging of session attributes



Bootstrapping the SP

- Choose your entityID

```
https://$HOSTNAME/shibboleth
```

- Should be:
 - Unique
 - Locally scoped
 - Logical representative
 - Unchanging
- Seen on the wire, configuration files, metadata, log files, etc

Bootstrapping the SP

- Relax some requirements, set your entityID and default IdP entityID

`$SHIB_CONF/shibboleth2.xml`

```
<Host name="$HOSTNAME" redirectToSSL="443">
```

```
<ApplicationDefaults entityID="https://$HOSTNAME/shibboleth"
REMOTE_USER="eppn persistent-id targeted-id" >
```

```
<SSO entityID="https://$WORKSH_IDP/saml2/idp/metadata.php" >
    SAML2 SAML1
</SSO>
```

```
<Handler type="Session" Location="/Session"
showAttributeValues="true"/>
```

Bootstrapping the SP

- Provide metadata remotely from your IdP

`$SHIB_CONF/shibboleth2.xml`

```
<MetadataProvider type="Chaining">  
<MetadataProvider type="XML"  
uri="https://$WORKSH_IDP/saml2/idp/metadata.php"  
backingFilePath="idp-metadata.xml" reloadInterval="3600"/>
```

- Backup at `$SHIB_RUN`

Uncomment whole `<MetadataProvider>`

Comment `<MetadataFilter>`

- Provide your SP's metadata to IdP
 - Metadata self-generated by your Service Provider
`https://$WORKSH_HOST/Shibboleth.sso/Metadata`

Bootstrapping the SP – Quick test

- Make sure configuration works

```
$ shibd -tc $SHIB_CONF/shibboleth2.xml
```

Service Provider reloads shibboleth2.xml automatically when it changes

- Try it with a browser

`https://$WORKSH_HOST/secure/`

`/secure/` is protected by `shibboleth2.xml` (`<RequestMap>`)

- Get session information

`https://$WORKSH_HOST/Shibboleth.sso/Session`
(you should see various attributes)

Bootstrapping SP - Logout

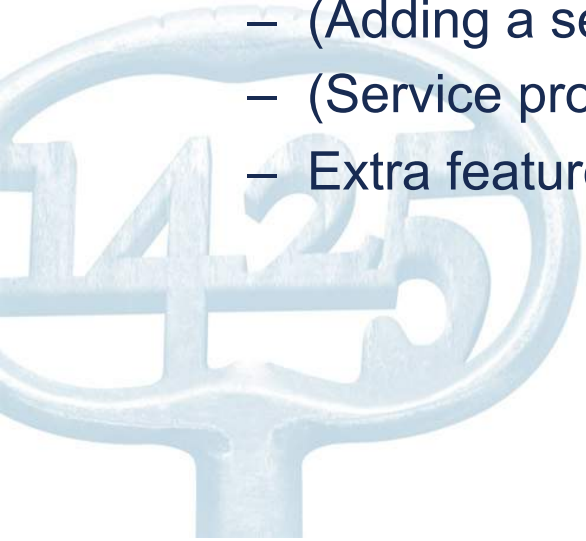
- Local logout
`https://$HOSTNAME/Shibboleth.sso/Logout`
This won't delete your session on the IdP!
- Close the browser in order to remove ALL your session cookies (~Firefox restore session functionality)
- Or delete session cookies using the browser or an extension, e.g.: Firefox Web Developer extension

→ Server side session remains and needs to timeout.



Program

- Installation
- Bootstrapping SP
- Configuration
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



Basic configuration

Goals:

1. Understand purpose and structure of SP configuration files
2. Increase log level to DEBUG
3. Configure metadata and add signature verification



Important directories

- `$SHIB_CONF`
 - Master and supporting configuration files
 - Locally maintained metadata files
 - HTML templates (customize them to adapt look&feel to your application)
 - Logging configuration files (*.logger)
 - Credentials (certificates and private keys)
- `$SHIB_RUN`
 - UNIX socket
 - Remotely fetched files (metadata, attribute-map)
- `$SHIB_LOG`
 - shibd.log & transaction.log
- `$WEB_LOG` (written by Shibboleth module)
 - native.log

Configuration files in \$SHIB_CONF

- **shibboleth2.xml** – main configuration file
- `apache*.config` – Apache module loading
- `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:

Adapting *.html files to match the look & feel of the protected application improves user experience.

shibboleth2.xml structure

Outer elements of the shibboleth2.xml configuration file

```
<OutOfProcess> / <InProcess>
<UnixListener> / <TCPLListener>
```

```
<StorageService>
<SessionCache>
<ReplayCache>
<ArtifactMap>
```

```
<RequestMapper>          Needed for session initiation and access control
```

```
<ApplicationDefaults>    Contains the most important settings of your SP
```


```
<SecurityPolicies>
```

ApplicationDefaults structure

You are most likely to change something in here:

- **<ApplicationDefaults>**
 - **<Sessions>** Defines handlers and how sessions are initiated and managed
 - **<Errors>** Used to display error messages. Provide here logo, e-mail and CSS
 - **<RelyingParty>** (*) To modify settings for certain IdPs/federations
 - **<MetadataProvider>** Defines the metadata to be used by the SP
 - **<TrustEngine>** Which mechanisms to use for signatures validation
 - **<AttributeExtractor>** Attribute map file to use
 - **<AttributeResolver>** Plugin for attribute querying
 - **<AttributeFilter>** Attribute filter file to use
 - **<CredentialResolver>** Defines certificate and private key to be use
 - **<ApplicationOverride>** (*) Can override any of the above for certain applications

Logging

- First thing to do in case of problems 
- shibd.log and transaction.log written by shibd, native.log written by Shibboleth module
- *.logger files contain predefined settings for output location and default logging level (INFO) along with useful categories to raise to DEBUG
- Log time is in UTC (~GMT)

Logging

- Raise categories

```
$ vim $SHIB_CONF/shibd.logger
```

```
log4j.rootCategory=DEBUG, shibd_log
```

- To implement *.logger changed:

```
$ touch shibboleth2.xml
```

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

- Try again `https://$WORKSH_HOST/secure/`

Metadata features

- Metadata describes the other components (IdPs) that the Service Provider can communicate with
- **Three primary methods built-in:**
 - Local file (you manage it)
 - Remote file (periodic refresh, local backup), i.e.
 - [https://\\$HOSTNAME/\\$PATH/saml2/sp/metadata.php](https://$HOSTNAME/$PATH/saml2/sp/metadata.php) (simplesamlphp SP)
 - [https://\\$HOSTNAME/Shibboleth.sso/Metadata](https://$HOSTNAME/Shibboleth.sso/Metadata) (Shibboleth SP)
 - [https://\\$HOSTNAME/\\$PATH/saml2/idp/metadata.php](https://$HOSTNAME/$PATH/saml2/idp/metadata.php) (simplesamlphp IdP)
 - [https://\\$HOSTNAME/idp/shibboleth](https://$HOSTNAME/idp/shibboleth) (Shibboleth IdP)
 - Dynamic resolution of entityID (=URL)
- Security comes from metadata filtering, either by you or the SP:
 - Signature verification
 - White and blacklists

Exercise: Signature verification

- The Test IdPs metadata is signed. Until now, it was loaded without checking, which is not secure and not recommended!
- First, increase security:
`$SHIB_CONF/shibboleth2.xml`

Uncomment MetadataFilter for signature verification:

```
<MetadataProvider type="XML" [...]  
uri="https://$WORKSH_IDP/idp/saml2/idp/metadata.php">  
  <MetadataFilter type="Signature" certificate="sp-cert.pem"/>  
</MetadataProvider>
```

Exercise: Signature verification cont'd

- Run

```
$ shibd -tc $SHIB_CONF/shibboleth2.xml
```

... and in the output you will see:

```
WARN OpenSAML.MetadataFilter.Signature [3]: filtering out
group at root of instance after failed signature check:
ERROR OpenSAML.Metadata.Chaining [3]: failure
initializing MetadataProvider: SignatureMetadataFilter
unable to verify signature at root of metadata
instance.
```

- Metadata could not be loaded because it was signed with a different key (we “broke” the setup). So, let’s get the right key...

Exercise: Signature verification cont'd

- Download the correct certificate
- Then fix it:
\$SHIB_CONF/shibboleth2.xml

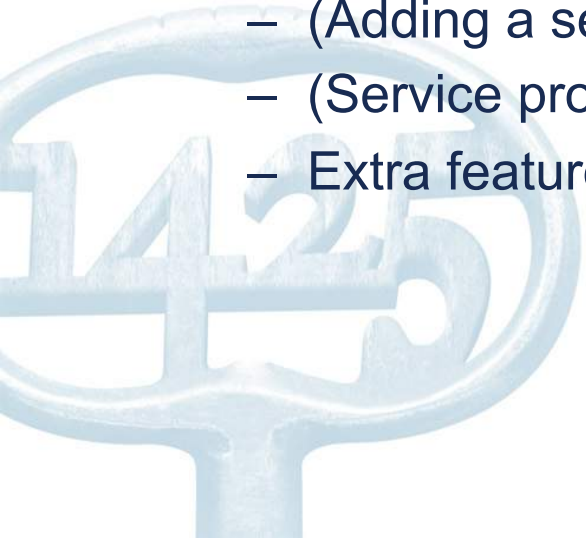
```
<MetadataProvider type="XML" [...] >  
  <MetadataFilter type="Signature"  
    certificate="$METADATA_SIGNING_CERTIFICATE"/>  
</MetadataProvider>
```

- Run again

```
$ shibd -tc $SHIB_CONF/shibboleth2.xml
```

Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



Attribute handling

Goals:

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



SP attribute terminology

- **Push**
Delivering attributes with SSO assertion via web browser
- **Pull**
Querying for attributes after SSO via back-channel (SP -> IdP)
- **Extraction**
Decoding SAML information into neutral data structures mapped to environment or header variables
- **Filtering**
Blocking invalid, unexpected, or unauthorized values based on application or community criteria
- **Resolution**
Resolving a SSO assertion into a set of additional attributes (e.g. queries)

Scoped attributes

- Common term for attributes that consist of a relation between a **value** and a **scope**, usually an organizational domain name

E.g. affiliation = “student@kuleuven.be”

- Makes values globally usable and unique
- Lots of special treatment in Shibboleth to make them more useful and "safe"
- Alternatively, split value and scope into separate attributes: affiliation=“student” and homeOrganization=“kuleuven.be”

Attribute mappings

- SAML attributes from any source are "**extracted**" using the configuration rules 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's and multiple attributes can be mapped to the same id
- The id can also be used as header name in the webserver for this attribute, not by default for Apache.

Dissecting an Advanced Attribute Rule

```
<Attribute id="affiliation" aliases="aff affil"  
  name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9">  
  <AttributeDecoder xsi:type="ScopedAttributeDecoder"  
    caseSensitive="false"/>  
</Attribute>
```

- `id`
The primary "id" to map into, also used in web server environment
- `aliases`
Optional alternate names to map into
- `name`
SAML attribute name or NameID format to map from
- `AttributeDecoder xsi:type`
Decoder plugin to use (defaults to simple/string)
- `caseSensitive`
How to compare values at runtime (defaults to true)

Adding attribute mappings

- Add first and lastname SAML 2 attribute mappings:
`$SHIB_CONF/attribute-map.xml`

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

- After saving, changes take effect immediately but **NOT** for any existing sessions
- Therefore, restart your browser (or delete your session cookies) and continue on next slide ...

Example: K.U.Leuven attribute mappings

- Attribute-map published on webserver, can be downloaded by all Shibboleth SP's
 - Warning: no possibility to sign → no verification possible
- \$SHIB_CONF/shibboleth2.xml

```
<!--  
<AttributeExtractor type="XML" validate="true" path="attribute-  
map.xml"/>  
-->  
<AttributeExtractor type="XML" uri="https://shib.kuleuven.be/  
download/sp/2.x/attribute-map.xml" backingFilePath="attribute-  
map.xml" reloadInterval="7200"/>
```

Example: K.U.Leuven attribute mappings

- Attribute-map made compatible with Shibboleth 1.3 (SAML1.1) naming conventions:
 - Old:

```
<Attribute name="urn:mace:dir:attribute-  
def:eduPersonScopedAffiliation" id="affiliation">  
    <AttributeDecoder xsi:type="ScopedAttributeDecoder"  
caseSensitive="false"/>  
</Attribute>
```

- New:

```
<Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9" id="affiliation">  
    <AttributeDecoder xsi:type="ScopedAttributeDecoder"  
caseSensitive="false"/>  
</Attribute>
```

Common identifiers

- **Local userid/netid/uid** (“intranet userid”), e.g. “u1234567”
Usually readable, persistent but not permanent, often reassigned, not unique
- **email address**, e.g. Foo.bar@kuleuven.be
Usually readable, persistent but not permanent, often reassigned, unique
- **eduPersonPrincipalName**, e.g. u1234567@kuleuven.be
Usually readable, persistent but not permanent, can be reassigned, unique
- **eduPersonTargetedID / SAML 2.0 persistent ID**
Not readable, semi-permanent, not reassigned, unique

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.
- IIS doesn't support setting REMOTE_USER

<https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeAccess>

Exercise: 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. "sn") as REMOTE_USER:
`$SHIB_CONF/shibboleth2.xml`
- `REMOTE_USER="sn eppn persistent-id targeted-id"`
- If sn attribute is available, it will be put into REMOTE_USER
 - Attribute sn has precedence over eppn in this case
 - This allows very easy "shibbolization" of some web applications

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 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

Default filter policy

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

Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features

Session initiation

Goals:

1. Learn how to initiate a Shibboleth session
2. Understand their advantages and 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

- .htaccess (dynamic) or apache2.conf (static)
- RequestMap
 - Requires Shibboleth to know exact hostname
 - Very powerful and flexible thanks to boolean/regex operations
- Try accessing `https://$HOSTNAME/`
You should get access because the directory is not protected

Content protection with .htaccess

- Prepare webserver (<Directory name="\$DOCROOT">)

```
AllowOverride AuthConfig
```

- Let's protect the directory by requiring a Shibboleth session:

```
$ mkdir $WEB_DOCROOT/secure2  
$ vim $WEB_DOCROOT/secure2/.htaccess
```

```
AuthType shibboleth  
ShibRequestSetting requireSession 1  
require valid-user
```

Test content protection rule

- Clear session and then access `https://$HOSTNAME/secure2`
- Authentication is enforced and access should be granted
- By now, all authenticated users get access
- Content protection with authorization will be covered later

Content protection with RequestMap

```
$ vim $WEB_DOCROOT/secure2/.htaccess
```

```
AuthType shibboleth  
require shibboleth
```

```
$SHIB_CONF/shibboleth2.xml
```

```
<Host name="$HOSTNAME" redirectToSSL="443">  
  <Path name="secure2" authType="shibboleth" requireSession="true"/>  
>  
</Host>
```

- Module (mod_shib) provides request URL to shibd to process it
- Clearing session and then accessing /secure2/ now, one also is forced to authenticate

RequestMap “Fragility”

- By default, Apache “trusts” the user’s web browser about what the requested hostname is and reports that value internally
- To illustrate the problem, try accessing this URL:
`https://$IP/secure2`

Script can be accessed unprotected/without a session... ?

- How to fix? Make Apache use configured ServerName
`apache2.conf`

```
UseCanonicalName On
```

Other content settings

- Requesting types of authentication
 - E.g enforce X.509 user certificate authentication
- Redirect to SSL
- 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
- **forceAuthn**
 - Disable Single-Sign on and force a re-authentication
- **isPassive**
 - Check whether a user has an SSO session and if he has, automatically create a session on SP without any user interaction
- Supplying a specific IdP to use for authentication

Lazy Sessions

- The mode of operation so far prevents an application from running without a login.
- Two other very common cases:
 - Public and private access to the same resources
 - Separation of application and SP session
- Semantics are:
if valid session exists
 - process it as usual (attributes in environment array, REMOTE_USER, etc.)



But if a session does NOT exist or is invalid, ignore it
and pass on control to webserver/scripts

Lazy Sessions example

- Construct URL

`https://$HOSTNAME/Shibboleth.sso/Login`

`?target=https://$HOSTNAME/Shibboleth.sso/Session`

- Shibboleth handler: `https://$HOSTNAME/Shibboleth.sso`
- Session Initiator: `/Login`
- Target location: `?target=https://$HOSTNAME/Shibboleth.sso/Session`
- Other options: `https://wiki.shibboleth.net/confluence/SHIB2/`
`NativeSPSessionCreationParameters`

- Most parameters can come from three places, in order of precedence:

- Query string parameter to Shibboleth handler
- A content setting (Webserver config or RequestMap)
- `<SessionInitiator>` element

Exercise: Lazy Sessions

```
$ vim $WEB_DOCROOT/secure3/.htaccess
```

```
AuthType shibboleth  
require shibboleth
```

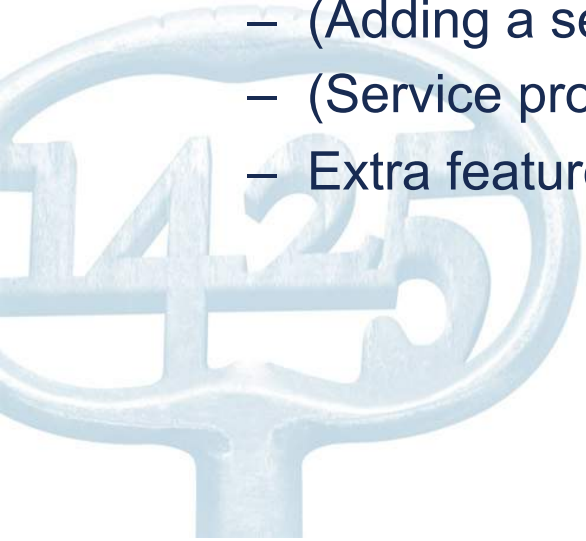
- **Save and extract PHP script from**
`http://shib.kuleuven.be/download/sp/test_scripts/
lazy_session.tar.gz`
at
`$WEB_DOCROOT/secure3/lazy_session.php`
Access `https://$HOSTNAME/secure3/lazy_session.php`

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
- Whole application with “lazy” Shibboleth session
 - Also allows for other authentication methods
 - Authorization 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 best solution**

Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



Access control

Goals:

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



Access control

- 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 webapplication



Access control

1.a apache2.conf	1.b .htaccess	2. XML AccessControl	3. Application Access Control
<ul style="list-style-type: none">▪ Easy to configure▪ Can also protect locations or virtual files▪ URL Regex	<ul style="list-style-type: none">▪ Dynamic▪ Easy to configure	<ul style="list-style-type: none">▪ Platform independent▪ Powerful boolean rules▪ URL Regex▪ Dynamic	<ul style="list-style-type: none">▪ Very flexible and powerful with arbitrarily complex rules▪ URL Regex Support
<ul style="list-style-type: none">▪ Only works for Apache▪ Not dynamic▪ Very limited rules	<ul style="list-style-type: none">▪ Only works for Apache▪ Only usable with “real” files and directories	<ul style="list-style-type: none">▪ XML editing▪ Configuration error can prevent SP from restarting	<ul style="list-style-type: none">▪ You have to implement it yourself▪ You have to maintain it yourself

1. Apache httpd.conf or .htaccess

- Work almost like known Apache “require” rules

```
require affiliation staff
```

```
require sn bar
```

- Special rules:
 - shibboleth (no authorization)
 - valid-user (require a session, but NOT identity)
 - user (REMOTE_USER as usual)
 - group (group files as usual)
 - authnContextClassRef, authnContextDeclRef
- Default is boolean “OR”, use ShibRequireAll for AND rule
- Regular expressions supported using special syntax:

```
require mail ~ ^.*@(icts|law).kuleuven.be$
```

Side note: Aliases

- If in the attribute-map.xml file, there is a definition like:

```
<Attribute
  name="urn:mace:dir:attribute-def:eduPersonAffiliation"
  id="Shib-EP-Affiliation"
  aliases="affiliation aff affil">
  [...]/>
```

- This allows using aliases in authorization rules, e.g.:

```
require affiliation staff
#instead of
require Shib-EP-Affiliation staff
```

- Aliases can also be used in RequestMap

1. Example .htaccess file

- Require a user to be staff member

`$WEB_DOCROOT/staff-only/.htaccess`

```
AuthType Shibboleth
ShibRequestSetting requireSession 1
require unscoped-affiliation staff
```

- Access

`https://$WORKSH_HOST/staff-only`
with user “**staff**”, access should be granted

- Try the same with user without “staff” attribute, access should be denied

1. Advanced .htaccess file

- Require a user to be a student **or** to have an entitlement:

```
$ mkdir $WEB_DOCROOT/student  
$ vim $WEB_DOCROOT/student/.htaccess
```

```
AuthType Shibboleth  
ShibRequestSetting requireSession 1  
require unscoped-affiliation student  
require entitlement ~ .*extra_access.*
```

Access:

`https://$WORKSH_HOST/student`

with user “**student**” and “**staff**”, access should be granted.

- Other users, access will be denied.

2. 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 can also be dynamically loaded from external file.

WARNING: Can bring down entire webserver

- Same special rules as .htaccess, adds boolean operators (AND,OR,NOT)

2. XML access control example

- Same as previous example but now with XML access control embedded in RequestMap

```
$ vim $DOCROOT/student/.htaccess
```

```
AuthType Shibboleth  
require shibboleth
```

```
$ vim $SHIB_CONF/shibboleth2.xml
```

```
<Host name="$WORKSH_HOST">  
  [...]   
  <Path name="student" authType="shibboleth" requireSession="true">  
    <AccessControl>  
      <OR>  
        <RuleRegex require="entitlement">.*extra_access.*</RuleRegex>  
        <Rule require="unscoped-affiliation">student</Rule>  
      </OR>  
    </AccessControl>  
  </Path>  
</Host>
```


3. 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/authorization

```
#PHP:
if ($_SERVER['affiliation'] == 'staff')
    { grantAccess() }

#Perl:
if ($ENV{'affiliation'} == 'staff')
    { &grantAccess() }

#ASP:
if (Request.ServerVariables('affiliation') == 'staff' ){
    { grantAccess() }
```

3. Application managed access control

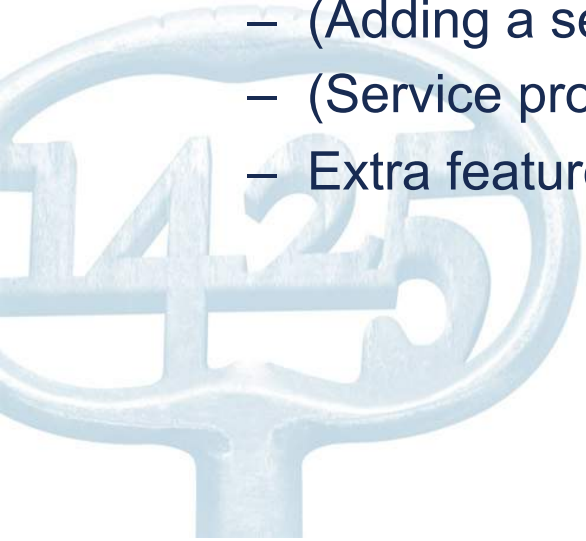
- Default is to use environment variables instead of HTTP headers (Apache)
 - Cannot be manipulated in any way from **outside**
- Unfortunately not all webserver support a mechanism to create custom variables within webserver (IIS, Sun/iPlanet)

Solution:

```
AuthType shibboleth
ShibRequestSetting requireSession 1
require shibboleth
ShibUseHeaders On
```

Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



Adding a separate (Shibboleth) application

Goals:

1. Define another application
2. Protect new application
3. Know how to configure them if necessary



Terminology

- **Service Provider** (physical)
 - An installation of the software on a server
- **Service Provider/"Resource"** (logical)
 - Web resources viewed externally as a unit
 - Each entityID identifies exactly one logical SP
- **SP Application**
 - Web resources viewed **internally** as a unit
 - Each applicationId identifies exactly one logical application
 - A user session is bound to exactly one application

Virtualization concepts

- **A single physical SP can host any number of logical SPs**
 - A logical SP can then include any number of "applications"
 - Web virtual hosting is often related but is also independent
 - Applications can inherit or override default configuration settings on a piecemeal basis
- **Multiple physical SPs can also act as a single logical SP**
 - Clustering for load balancing and failover




Adding an application

- **Goal:** Add a second application with a different entityID living in its own virtual host

`$SHIB_CONF/shibboleth2.xml`

```
<RequestMap applicationId="default">
  <Host name="$WORKSH_HOST2" applicationId="alt"/>

[...]
```



```
  <ApplicationOverride id="alt" entityID="https://$WORKSH_HOST2"/>
</ApplicationDefaults>
```

Adding an application

- Test application:

`https://$WORKSH_HOST2/secure`

- The IdP will throw an **ERROR** (entityID is not trusted)

Error Message: SAML 2 SSO profile is not configured for relying party 'https://\$WORKSH_HOST2'

- Check logging `$SHIB_LOG/shibd.log` and `$WEB_LOG/native.log` (DEBUG)

You should see the new entityID



Adding an application

- `<ApplicationOverride>`

Rule of thumb is that any settings you don't override inside the element will be inherited from the `<ApplicationDefaults>` element that surrounds the override .

- Limitations:

You have to supply all the settings needed in the `<Sessions>` element because of the need to override the handlerURL.

You do **NOT** have to redefine all of the handler child elements.

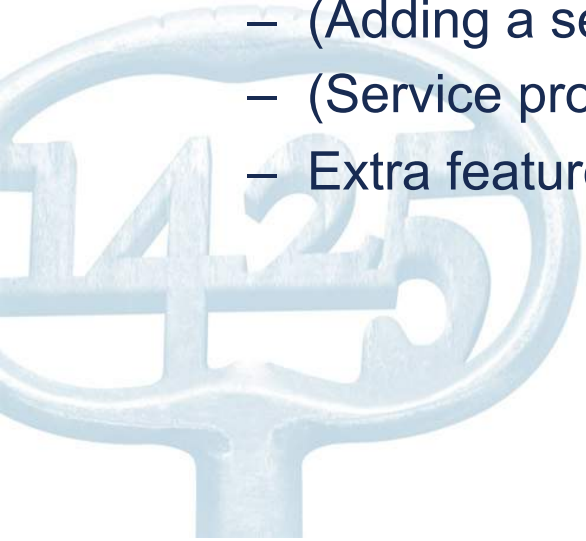
- The handlerURL **MUST** be unique for each SP and **MUST** map to the same applicationId
- Respect the XML sequence!

- Configure multiple physical installations to share an entityID, and possibly credentials
- Configuration files often can be identical across servers that share an external hostname
- Session management:
 - SP itself now clusterable via ODBC or memcached
 - Host shibboleth service on one system



Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



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)
 - E.g. `/Shibboleth.sso/Login`
 - AssertionConsumerService (incoming SAML response)
 - E.g. `/Shibboleth.sso/SAML/POST`
 - LogoutInitiator (SP signout)
 - E.g. `/Shibboleth.sso/Logout`
 - SingleLogoutService (incoming SLO)
 - ManageNameIDService (advanced SAML)
 - ArtifactResolutionService (advanced SAML)
 - Generic (diagnostics, other useful features)
 - E.g. `/Shibboleth.sso/Session`
 - `/Shibboleth.sso/Status`
 - `/Shibboleth.sso/Metadata`

SP handlers

- The URL of a handler = handlerURL + the Location of the handler.
 - e.g. for a virtual host *testsp.example.org* with handlerURL of *"/Shibboleth.sso"*, a handler with a Location of *"/Login"* will be <https://testsp.example.org/Shibboleth.sso/Login>
- Handlers aren't always SSL-only, but usually should be (handlerSSL="true").
- Metadata basically consists of entityID, keys and handlers
- Handlers are never "protected" by the SP
 - But sometimes by IP address (e.g. with `ac1="127.0.0.1"`)

Program

- Installation
- Bootstrapping SP
- **Configuration**
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features

- Extra layer of security
- Request reauthentication by the IdP
- Configuration:
 - forceAuthn
 - Content settings (RequestMap, SessionInitiator, .htaccess)
 - query string parameter (overrides every previous setting)
- Easily overwritten → implement check if honored
 - maxTimeSinceAuthn
 - authnContextClassRef == urn:oasis:names:tc:SAML:2.0:ac:classes>PasswordProtectedTransport

→ Try It! 😊

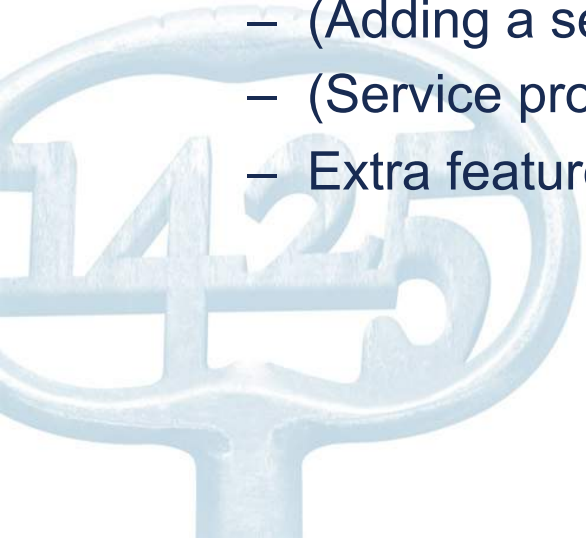
Passive logon

- SP tries to login user without user interaction
- User interaction needed → logon fails quietly
- Send a passive authentication request
- Lazy session
- Javascript to initiate session and 'remember' (set cookie) if failed
- Configuration:
 - Content settings (RequestMap, SessionInitiator .htaccess)
 - query string parameter (overrides every previous setting)

→ Try It! ☺

Program

- Installation
- Bootstrapping SP
- Configuration
 - Basic configuration
 - Attribute handling
 - Session Initiation
 - Access control
 - (Adding a separate application)
 - (Service provider handlers)
 - Extra features



Questions

