That's the University Slide
Overview

- Introducing the SPIE Project
- Integrating Shibboleth into Applications  
  – Demos of Applications using JAAS
- Integrating ARPs into Applications  
  – Demo of SP-side ARP
- Using Privilege Management with Shibboleth  
  – Demo using PERMIS
Introducing the SPIE Project

• Overall Aim
  “…investigate seamless authentication and authorisation access across information environments”

• Oxford: Institutional Context
  – Access to Local and Remote Protected Resources
  – Integration with Local Infrastructure
  – Integration with Institutional Applications
Approaches of Integrating Shibboleth 1.3c (Java & C-SP)

• Using a Filter (mod_shib, shib-filter)
• Using Attribute Assertions in
  – the Application
  – the Application's Security Framework
  – the Application Server (container-level)
  – any combination of the above
Filter approach
(string matching URLs)

A user Request

http://uni.ac.uk/staffonly/

And/or a Username/Attribute

eduPersonScopedAffiliation staff@test.edu
e等。

Is matched with a String

require affiliation staff@test.edu staff@other.edu ...

Or a Regular Expression

require affiliation ~ ^staff@.+\.[edu]$
Filter approach (using URLs)

Internet user

entitlement: library

http://uni.ac.uk/staffonly/

http://uni.ac.uk/library/

http://uni.ac.uk/finance/

http://uni.ac.uk/exampapers/
Filter approach (using URLs)

\[
\text{if (entitlement:library) allow "/library"}
\]

http://uni.ac.uk/staffonly/

http://uni.ac.uk/library/

http://uni.ac.uk/finance/

http://uni.ac.uk/exampapers/
Filter approach (using URLs)

if (entitlement:library) allow "/library*"

http://uni.ac.uk/staffonly/
http://uni.ac.uk/library/
http://uni.ac.uk/library/../../exams/
http://uni.ac.uk/finance/
http://uni.ac.uk/exams/
Filter Approach
(Greyhound Racing)

User Tom
maxiumBet: 500

etc.
if (maxiumBet:500) allow “/placebet?amount=500”
if (maxiumBet:500) allow “/placebet?amount=499”
if (maxiumBet:500) allow “/placebet?amount=498”
if (maxiumBet:500) allow “/placebet?amount=497”

etc.
Filter Approach

• Good for protecting Static Data
  – or for messy webapps you don't want to touch

• But
  – Easy to make mistakes in matching rules
  – Not appropriate for protecting dynamic content
  – Difficult to integrate with Privilege Management
Using Attributes in the Application

- User's name (and attributes) checked against application's security configuration
  - configuration has to be done (machines * applications) times
  - difficult to maintain and change
Using attributes in the application
Example for Application-based Security

Login using University card barcode
Example for application-based security

- Gives access to restricted facilities
- Using SQL db for authZ
Using Attributes in the Application's Security Framework
Using attributes in the application's security framework

<table>
<thead>
<tr>
<th>Att. Name</th>
<th>Att. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available attributes:</td>
<td></td>
</tr>
<tr>
<td>urn:mae:dina:attribute-def:eduPersonAffiliation</td>
<td>member</td>
</tr>
<tr>
<td>urn:mae:dina:attribute-def:eduPersonScopedAffiliation</td>
<td>member</td>
</tr>
<tr>
<td>urn:mae:dina:attribute-def:eduPersonTargetedID</td>
<td>TXXFHJ2Qnt60QAYCq5B9dVXrK8U=</td>
</tr>
<tr>
<td>urn:mae:dina:attribute-def:eduPersonPrincipalName</td>
<td>chrtsf</td>
</tr>
<tr>
<td>urn:mae:dina:attribute-def:eduPersonEntitlement</td>
<td>webauth</td>
</tr>
<tr>
<td>uPortalTemplateUserName</td>
<td>demo</td>
</tr>
<tr>
<td>username</td>
<td>chrtsf</td>
</tr>
<tr>
<td>Unavailable attributes:</td>
<td></td>
</tr>
<tr>
<td>displayName</td>
<td>[Not available]</td>
</tr>
<tr>
<td>givenName</td>
<td>[Not available]</td>
</tr>
<tr>
<td>mail</td>
<td>[Not available]</td>
</tr>
<tr>
<td>sn</td>
<td>[Not available]</td>
</tr>
<tr>
<td>user.home-info.online.email</td>
<td>[Not available]</td>
</tr>
<tr>
<td>userName.family</td>
<td>[Not available]</td>
</tr>
<tr>
<td>userName.given</td>
<td>[Not available]</td>
</tr>
</tbody>
</table>
Using Attributes in the Application's Security Framework
Using Attributes in the Application Server

- Apps can use container's API

Identity Management

Container's Security Framework

App 2
Application
App n

Container's API

shib
krb5
ldap
...

Apps can use container's API
Using Attributes in the Application Server (example Servlet-API)

```java
Servlet API based AuthN/AuthZ

Your shibboleth attributes are:

```
urn:mace:dir:attribute-def:eduPersonAffiliation member
urn:mace:dir:attribute-def:eduPersonScopedAffiliation member
urn:mace:dir:attribute-def:eduPersonTargetedID qVPGCFXvaY79YpFS7o6e99NOFLI=
urn:mace:dir:attribute-def:cn chrisf
urn:mace:dir:attribute-def:eduPersonPrincipalName chrisf
urn:mace:dir:attribute-def:eduPersonEntitlement [autocreated, webauth, user, warpeadmin]
```

```java
request.getRemoteUser()

request.getRemoteUser() returns: chrisf

request.isUserInRole()

using isUserInRole() to check group membership, as defined in META-INF/web.xml

Yes! You are member of role member

Check for role membership: [member] [Check]
```
Using Attributes in the Application Server (example Servlet-API)
Summary of Application Integration

• System Management Tools / smaller Apps
  – directly use attributes in code
  – authZ rules hardcoded, changes difficult
  – security and applications concerns difficult to separate

• Larger apps (portals, weblearn, ...)
  – typically make use of a security framework
  – often can be shibbolized with no or very minor changes
Our approach to integrate Java Applications with Shibboleth

• Create a reusable security module that can be used with existing applications
• Store Attribute Release Policy together with user attributes in a LDAP directory
• Allow users to change their ARPs (currently simplistic jsp-based GUI)
• Allow applications to request changes to the ARP and attributes
Shibboleth – Java integration with SpieJaas

- In Java, the security modules typically rely on JAAS
- JAAS is a Java standard for pluggable security modules
Shibboleth – Java integration with SpieJaas

1-6) Normal Shibboleth protocol run
7) User's attributes sent to SpieJaas module
8) Normal login request sent to application
9) User is logged in using Shibboleth
SP-side ARP editor

This service requests the following attributes:

urn:mace:dir:attribute-def:eduPersonEntitlement
urn:mace:dir:attribute-def:eduPersonAffiliation
urn:mace:dir:attribute-def:eduPersonTargetedID

Do you agree to release these attributes?

[OK] Remember my choice
Please select the user you want to edit:

- ball0346
- chrisf
- deepu
- default
- demo
- ecjet
- ecjet2
- ecjet3
- fq
- larry
- manc0529
- math0162
- mats
- mike
- mynewuser
- newc1666
- pemb2021
- phys0102
- scro0663
- scro0826
- some0162
- template
- test1
- test2
- test3
- test4
- test5
- test6
- test7
- test8
- univ1985
- worc2128
- wycl0564

If you don't already have a username, you can create one here:

name: [ ] create
WARPP(e)
Webservice ARP (e)ditor

• Very simple WebService API
• Uses Shib-resolver for read access
  – All (future) attribute plug-ins can be used
• Uses JNDI for write access
• LdapArpRepository plug-in creates ARP on the IdP
N-Tier authN with Shibboleth 1.3

- Shibbolize an AuthN System, not the Application

- currently working to do the same with WebAuth (Kerberos)
Open Problems

• How can the application talk-back to the security services (e.g. request higher assurance level)