

National e-Science Centre

PERMIS Tutorial
Grid Computing module
21st February 2006

University of Glasgow, Scotland

Anthony Stell



UNIVERSITY of GLASGOW

National e-Science Centre

Authorization

- OK, I know who you are (authn) but, based on that identity, what are you allowed to do (authz)?
- Saying simply “yes” or “no” to one request is easy…
- Saying “yes, but…” is more difficult (i.e. adding conditions to the “yes”)…
- We want people to be able to use resources outside their domain but still keep the sensitive stuff protected…

Globus – not good enough…

- Grid-map files just don't cut it:
  - Essentially a list of user accounts versus unique DNs (Distinguished Names)
  - Unwieldy to maintain => not scaleable
  - Very "coarse-grained" security
- Other authz methods out there:
  - CAS (Community Authorization Service)
  - VOMS (Virtual Organisation Membership Service)
  - Many more, all of them have their good and bad points…
- Globus defined a "call-out"…
  - This is an API in the Globus Toolkit to let some other third-party do the authz dirty work…

Security Assertion Markup Language (SAML)

National e-Science Centre

- Standard language of security
- Defined by the OASIS standards body
- Widely accepted as the language to use for passing security assertions between grid elements…
- When Globus “calls out”, this is the language it is using to do so…

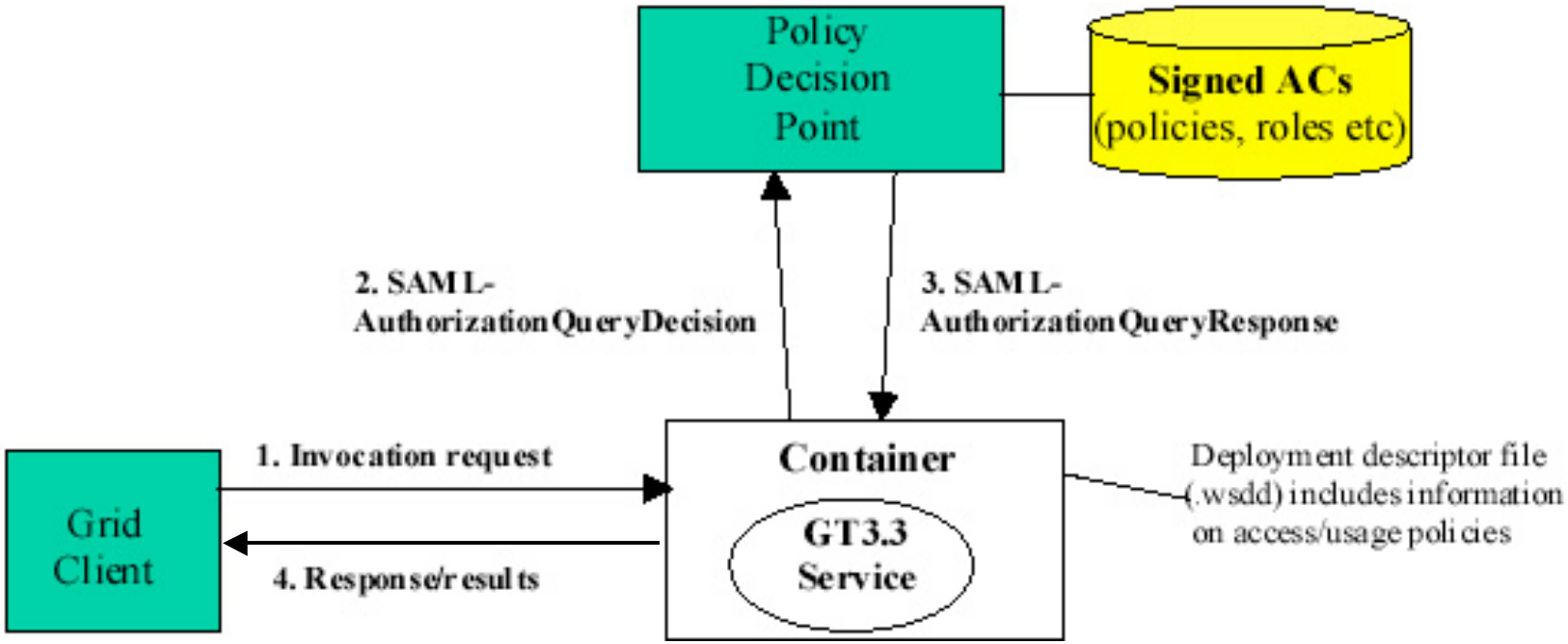
PERMIS

- "PrivilEge and Role Management Infrastructure and Standards validation"
- Essentially a module that interprets SAML
- Takes a SAML assertion from a grid service (coded using Globus Toolkit) and compares against a policy.
- Returns an authorization decision
- Uses "role-based" access control
- More than a look-up table – it defines a Privilege Management Infrastructure (PMI)

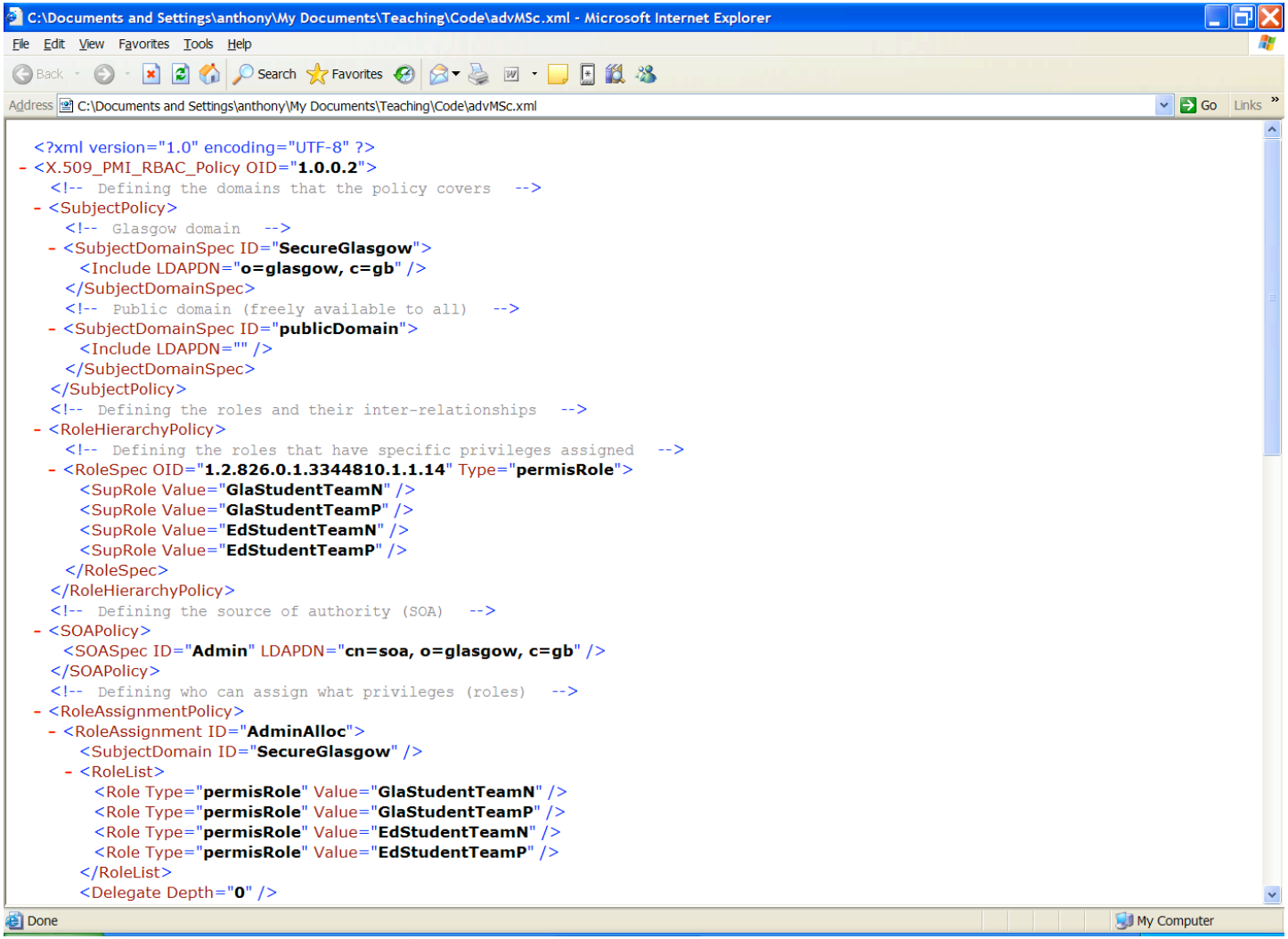
PKI vs. PMI

Concept	PKI Entity	PMI Entity
Certificate	Public Key Certificate (PKC)	Attribute Certificate (AC)
Certificate Issuer	Certification Authority (CA)	Attribute Authority (AA)
Certificate User	Subject	Holder
Certificate Binding	Subject's name to Public Key	Holder's Name to Privilege Attribute(s)
Revocation	Certificate Revocation List (CRL)	Attribute Certificate Revocation List (ACRL)
Root of trust	Root Certification Authority or Trust Anchor	Source of Authority (SOA)
Subordinate authority	Subordinate Certification Authority	Attribute Authority (AA)

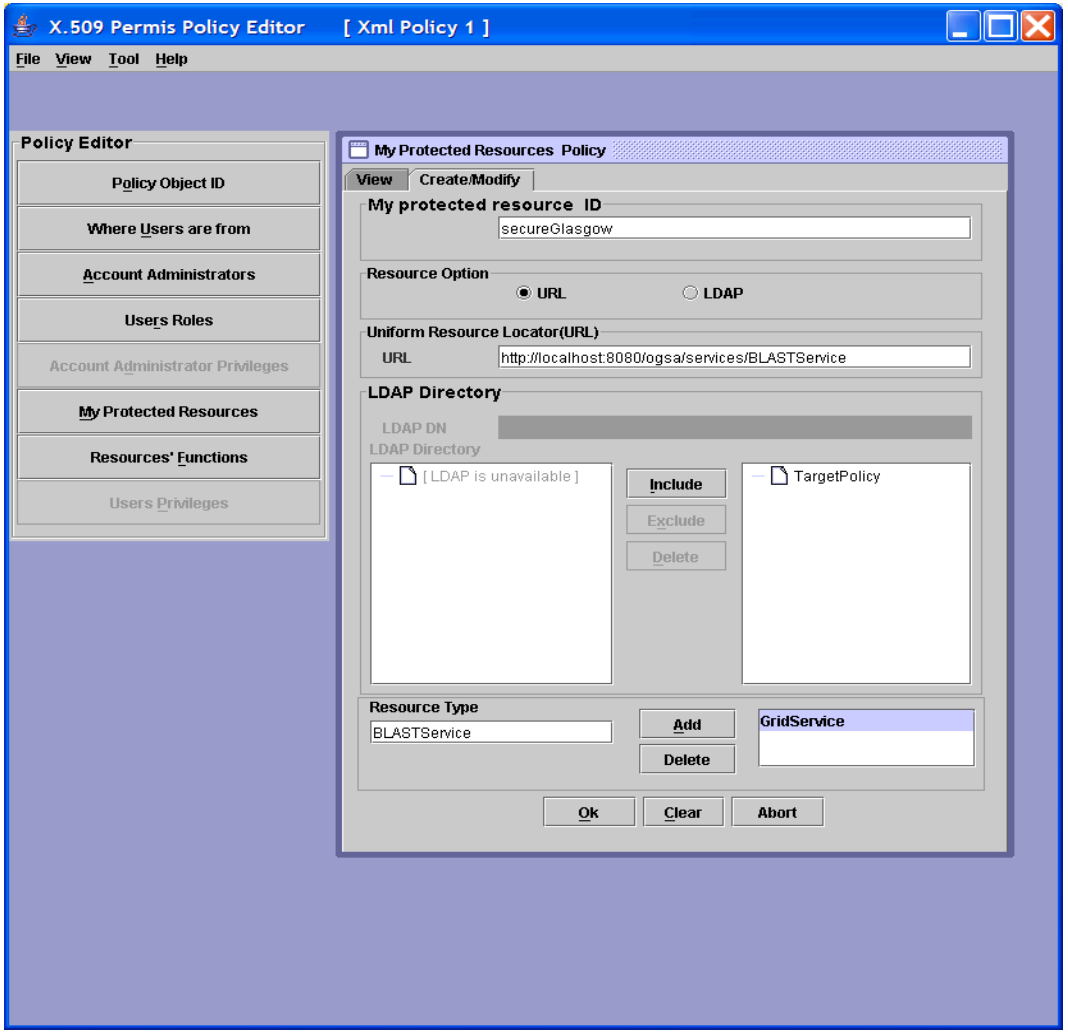
Architecture



XML Policy


C:\Documents and Settings\anthony\My Documents\Teaching\Code\advMSc.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\anthony\My Documents\Teaching\Code\advMSc.xml
Go Links
<?xml version="1.0" encoding="UTF-8" ?>
<X.509_PMI_RBAC_Policy OID="1.0.0.2">
<!-- Defining the domains that the policy covers -->
<SubjectPolicy>
<!-- Glasgow domain -->
<SubjectDomainSpec ID="SecureGlasgow">
<Include LDAPDN="o=glasgow, c=gb" />
</SubjectDomainSpec>
<!-- Public domain (freely available to all) -->
<SubjectDomainSpec ID="publicDomain">
<Include LDAPDN="" />
</SubjectDomainSpec>
</SubjectPolicy>
<!-- Defining the roles and their inter-relationships -->
<RoleHierarchyPolicy>
<!-- Defining the roles that have specific privileges assigned -->
<RoleSpec OID="1.2.826.0.1.3344810.1.1.14" Type="permisRole">
<SupRole Value="GlaStudentTeamN" />
<SupRole Value="GlaStudentTeamP" />
<SupRole Value="EdStudentTeamN" />
<SupRole Value="EdStudentTeamP" />
</RoleSpec>
</RoleHierarchyPolicy>
<!-- Defining the source of authority (SOA) -->
<SOAPolicy>
<SOASpec ID="Admin" LDAPDN="cn=soa, o=glasgow, c=gb" />
</SOAPolicy>
<!-- Defining who can assign what privileges (roles) -->
<RoleAssignmentPolicy>
<RoleAssignment ID="AdminAlloc">
<SubjectDomain ID="SecureGlasgow" />
<RoleList>
<Role Type="permisRole" Value="GlaStudentTeamN" />
<Role Type="permisRole" Value="GlaStudentTeamP" />
<Role Type="permisRole" Value="EdStudentTeamN" />
<Role Type="permisRole" Value="EdStudentTeamP" />
</RoleList>
<Delegate Depth="0" />
Done
My Computer

PERMIS Tools: Policy Editor



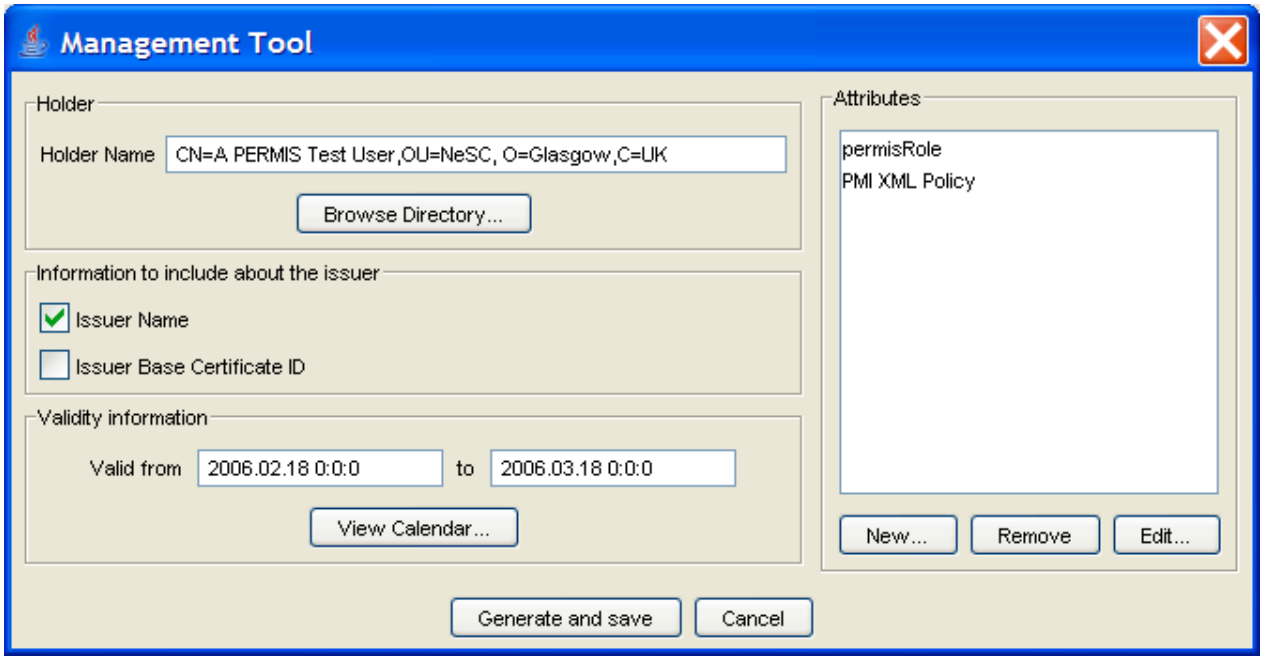
Target Access Policy

```
<TargetAccessPolicy>
        <TargetAccess ID="studentTeamNAccess">
                <RoleList>
                        <Role Type="permisRole" Value="GlaStudentTeamN"/>
                </RoleList>
                <TargetList>
                        <Target Actions="runBLASTN">
                                <TargetDomain ID="BlastService"/>
                        </Target>
                </TargetList>
        </TargetAccess>
        <TargetAccess ID="studentTeamPAccess">
                <RoleList>
                        <Role Type="permisRole" Value="GlaStudentTeamP"/>
                </RoleList>
                <TargetList>
                        <Target Actions="runBLASTP">
                                <TargetDomain ID="BlastService"/>
                        </Target>
                </TargetList>
        </TargetAccess>
</TargetAccessPolicy>
```

PERMIS Tools: Privilege Allocator

National e-Science Centre

- **Tool to create signed Attribute Certificates:**
  - **Standard format, stored in an LDAP directory**
  - **Can store roles and/or XML policies**



Under the hood…

- LDAP
  - Back-end directory
- OpenSSL
  - Used to set up the PKI certificates that PERMIS necessarily uses.
- XML
  - Used to describe policy – nice and hierarchical
- GSI
  - Used to secure the calls between service and policy engine

Limitations

- **Method-only execution**
  - My personal bug-bear…
  - Can only run a method, which will then say "yes" or bomb out with an authorization exception
- **Big overhead of supporting infrastructure**
  - Need to appreciate the niceties of Globus, OpenSSL, LDAP…
  - Lots of scope for things to go wrong…
- **Not a mature technology yet**
  - But nothing in Grid really is…

In practice (1)…

- **Requires tweaking your own service description:**
  - **Specify that you're going to do authz in another way to that provided by Globus:**
    - <parameter="authorization" value="custom"/>
  - **Point your service to the authorization service:**
    - <parameter="authzService"
    - value="http://localhost:8080/ogsa/services/decider/PermisAuthorizationService"/>
  - **Point your service to the Java class that implements this authz service:**
    - <parameter="authzClass"
    - value = "org.globus.ogsa.impl.security.authorization.SAMLAuthorizationCallout"/>

In practice (2)…

- Set up the PERMIS service to look for the appropriate infrastructure:
  - LDAP
  - Policy OID (Object IDentifier)
  - SOA (Source Of Authority)
- Set your environment variables
- Extract your key and certificate from your credentials (already done for you)
- Switch GRIM (Grid Resource Identity Mapper) security handles off
- Won't go into details here – all in assignment

In practice (3)…

- **Need to add GSI code to your client:**
  - **Specify that you're verifying the integrity of the message:**
    - ((Stub) *stubname*)._setProperty(Constants.GSI_SEC_CONV, Constants.SIGNATURE);
  - **Specify that this won't be accepted from any other delegated party:**
    - ((Stub) *stubname*)._setProperty(GSIConstants.GSI_MODE, GSIConstants.GSI_MODE_NO_DELEG);
  - **Specify that no *client* authorization will be performed:**
    - ((Stub) *stubname*)._setProperty(Constants.AUTHORIZATION, NoAuthorization.getInstance());

Error Messages: Globus Client

- "Connection Refused"
  - You haven't started your container
    - globus-start-container
- "Proxy file not found"
  - You haven't created your proxy file
    - grid-proxy-init

Error Messages: GSI

- **"Unknown CA"**
  - **Credentials appear to come from a CA that GSI can't verify (several possibilities):**
    - Remove all references to "grim" in server-config.wsdd under $GLOBUS_LOCATION (i.e. run a grep within the file and comment out all lines highlighted)
    - Check that the root certificate (fcec9e22.0) and signing policy (fcec9e22.signing_policy) are present and correct in:
      - ~/.globus/certificates
      - /home/globus/.globus/certificates
      - /etc/grid-security/certificates
    - Check that the certificates are valid
      - **grid-cert-info**

Error Messages: GSI

- "Hashes do not verify"
  - Signature check thinks that the sender is not who they say they are:
    - Change the line that refers to Constants.SIGNATURE in your client to Constants.ENCRYPTION

Error Messages: PERMIS

- **"Target is out of Target Domain"**
  - **Your service name does not match the target specified in the policy**
    - Service name should appear in your container as:
      - http://localhost:8080/ogsa/services/BLASTService
    - Tweak the service name in the server-config.wsdd in your own build tree then re-deploy into Globus container to sort this out. It should appear as simply "BLASTService"

Success Message: PERMIS

- Unbelievably, this means that it works:

```
ajstell@labpc-5: ~/old_services/grid_services/RossService
faultSubcode:
faultString: org.globus.ogsa.impl.security.authorization.AuthorizationException: Authorization failed.
faultActor:
faultNode:
faultDetail:
        {http://xml.apache.org/axis/}stackTrace: org.globus.ogsa.impl.security.authorization.AuthorizationException: Authorization failed
.
        at org.globus.ogsa.impl.security.authorization.SAMLAuthorizationCallout.authorize(SAMLAuthorizationCallout.java:355)
        at org.globus.ogsa.impl.security.authorization.AuthorizationHandler.invoke(AuthorizationHandler.java:111)
        at org.apache.axis.strategies.InvocationStrategy.visit(InvocationStrategy.java:71)
        at org.apache.axis.SimpleChain.doVisiting(SimpleChain.java:150)
        at org.apache.axis.SimpleChain.invoke(SimpleChain.java:120)
        at org.apache.axis.server.AxisServer.invoke(AxisServer.java:287)
        at org.globus.ogsa.server.ServiceThread.doPost(ServiceThread.java:438)
        at org.globus.ogsa.server.ServiceThread.process(ServiceThread.java:287)
        at org.globus.ogsa.server.ServiceThread.run(ServiceThread.java:185)


org.globus.ogsa.impl.security.authorization.AuthorizationException: Authorization failed.
        at org.apache.axis.message.SOAPFaultBuilder.createFault(SOAPFaultBuilder.java:260)
        at org.apache.axis.message.SOAPFaultBuilder.endElement(SOAPFaultBuilder.java:169)
        at org.apache.axis.encoding.DeserializationContextImpl.endElement(DeserializationContextImpl.java:1083)
        at org.apache.xerces.parsers.AbstractSAXParser.endElement(Unknown Source)
        at org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanEndElement(Unknown Source)
        at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source)
        at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source)
        at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source)
        at org.apache.xerces.parsers.DTDConfiguration.parse(Unknown Source)
        at org.apache.xerces.parsers.XMLParser.parse(Unknown Source)
        at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source)
        at javax.xml.parsers.SAXParser.parse(SAXParser.java:345)
        at org.apache.axis.encoding.DeserializationContextImpl.parse(DeserializationContextImpl.java:252)
        at org.apache.axis.SOAPPart.getAsSOAPEnvelope(SOAPPart.java:538)
        at org.apache.axis.Message.getSOAPEnvelope(Message.java:376)
        at org.globus.ogsa.handlers.RoutingHandler.getRouterHeader(RoutingHandler.java:31)
        at org.globus.ogsa.handlers.RoutingResponseHandler.invoke(RoutingResponseHandler.java:26)
        at org.apache.axis.strategies.InvocationStrategy.visit(InvocationStrategy.java:71)
        at org.apache.axis.SimpleChain.doVisiting(SimpleChain.java:150)
        at org.apache.axis.SimpleChain.invoke(SimpleChain.java:120)
        at org.apache.axis.client.AxisClient.invoke(AxisClient.java:193)
        at org.apache.axis.client.Call.invokeEngine(Call.java:2564)
        at org.apache.axis.client.Call.invoke(Call.java:2553)
        at org.apache.axis.client.Call.invoke(Call.java:2248)
        at org.apache.axis.client.Call.invoke(Call.java:2171)
        at org.apache.axis.client.Call.invoke(Call.java:1691)
        at org.globus.gcAssignment.stubs.SearchSortService.bindings.SearchSortGridServiceSOAPBindingStub.sortMethod(SearchSortGridService
SOAPBindingStub.java:796)
        at permis.Client.sort(Client.java:111)
        at permis.Client.main(Client.java:51)
```

Success Message: PERMIS

- Key phrase to look for is:
  - "AuthorizationException: Authorization Failed"
- And on the server side:
  - org.globus.ogsa.impl.security.authorization.SAMLAuthorizationCallout
  - ERROR: Authorization failed.
- But remember it should work according to your use cases…

Demo

- Last year's assignment – slightly different to this year:
  - Application to search and sort the complete works of Shakespeare – wrapped as a PERMIS-protected grid service
- The user should be able to run the search service but not the sort service…

PERMIS Resources

- PERMIS Home page:
  - http://sec.isi.salford.ac.uk/permis
- NeSC support pages ("Grid Security" section):
  - http://labserv.nesc.gla.ac.uk/projects/etf
- Module hand-out instructions:
  - http://www.dcs.gla.ac.uk/~ajstell/tutorials/permis.html
- My email:
  - ajstell@dcs.gla.ac.uk
- For enthusiasts (i.e. not examinable):
  - GT4/PERMIS integration guide
  - http://labserv.nesc.gla.ac.uk/projects/etf/gt4howto/permis.html