Grid Computing

Security - Authorization
Community Authorization and Privilege Management

Paul A. Farrell
Fall 2006

Authentication & Authorization

- Identity established through authentication
  - No info on user permissions/rights/privilege
  - A separate infrastructure is needed to manage user privilege

- Authorisation is an ongoing research area with many solutions
  - Most solutions involve integrating many separate technologies
    - And often many AuthZ techniques

So many models…

- Various technologies for authorization including
  - PERMIS
    - PriviEge and Role Management Infrastructure Standards Validation
      - http://www.permis.org
  - VOMS
  - Community Authorisation Service
    - http://www.globus.org/security/CAS/
  - AKENTI
  - CARDEA
  - All of them predominantly work at the local policy level

Access Control Lists (ACLs)

- Lets start with the simplest scenario:
  - Once a user has authenticated they are checked against a local list of users
  - Simple to understand and works well for mini-grids
  - The generic AuthZ for GSI (Grid Security Infrastructure) is an ACL
    - Will show you one in the next section
    - But remember they have defined the GGF API to use with ANY AuthZ

- But...
  - What if access to a resource is needed for a different purpose by the same person?
    - Multiple entries or multiple lists?
  - What if we want HUNDREDS of users?
    - BUSY, BUSY sys admins!
A better way…

- Just a straight list of users is too difficult to maintain and is not flexible enough for Grids

- What defines a persons permissions on a resource usually?
  - What kind of jobs do people have?
    - Doctor, Nurse, Student, Lecturer, Director, CEO, SysAdmin, PhD
  - People come and go but job descriptions generally are static
    - Any exceptions should be easy to manage

Role Based Access Control

- Access to a resource should be granted according to a user’s ROLE within the VO
  - Multiple Roles may be held by a user
  - Different levels of AuthZ may be enforced.
  - Role hierarchies may be supported
  - Access may be granted by Role only
    - If anonymous access is required
    - No policy changes required as users come and go
    - Happy sys admins!
    - Just grant them the necessary role when they join the VO and they will have access...

  - So how do we grant roles to users?

Privilege Management Infrastructures (PMIs)

- We can utilise the secure infrastructure provided by X.509 certificates to assign roles to users
  - We need an extension to the X.509 specification to support PRIVILEGE ATTRIBUTES
  - So as well as the normal info in their certificate, a user may be assigned one or more ATTRIBUTE CERTIFICATES which contain a signed assertion of their role within the VO
  - Many similarities to PKIs…

PKI and PMI

- A PMI is to authorisation what a PKI is to authentication – hence similar concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>PKI Entity</th>
<th>PMI Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate</td>
<td>Public Key Certificate (PKC)</td>
<td>Attribute Certificate (AC)</td>
</tr>
<tr>
<td>Certificate Issuer</td>
<td>Certification Authority (CA)</td>
<td>Attribute Authority (AA)</td>
</tr>
<tr>
<td>Certificate User</td>
<td>Subject</td>
<td>Holder</td>
</tr>
<tr>
<td>Certificate Binding</td>
<td>Subject’s name to Public Key</td>
<td>Holder’s Name to Privilege Attribute(s)</td>
</tr>
<tr>
<td>Revocation</td>
<td>Certificate Revocation List (CRL)</td>
<td>Attribute Certificate Revocation List (ACRL)</td>
</tr>
<tr>
<td>Root of trust</td>
<td>Root Certification Authority or Trust Anchor</td>
<td>Source of Authority (SOA)</td>
</tr>
<tr>
<td>Subordinate authority</td>
<td>Subordinate Certification Authority</td>
<td>Attribute Authority (AA)</td>
</tr>
</tbody>
</table>
Generic Authorisation

- A generic framework for authorisation is defined in X.812 ISO 10181-3 Acc. Ctrl. Framework

Grid APIs for Generic Authorisation

- SAML AuthZ specification provides generic PEP approach for Grid services
  - ... or at least all GT3.3+ based services

Access Control Policies

- Basic idea is to define:
  - roles applicable to specific VO
  - actions allowed/not allowed for VO members
  - resources comprising VO infrastructure (computers, data resources etc)

  A policy then consists of sets of these rules:
  - \{ Role x Action x Target \}
  - Can user with VO role X invoke service Y on resource Z?
  - Policy itself can be represented in many ways,
    - e.g. XML document, SAML, XACML, ...
CAS – Community Authorization Service

- Resource providers specify a coarse-grained policy
  - They grant the privileges to the community
  - Resource providers are spared day-to-day policy administration tasks (e.g., adding and deleting users, modifying user privileges).
- Fine grained policy decisions delegated to the community administrator of the community served by CAS
  - Resource providers have established a trust relationship with this administrator
  - The Administrator uses CAS to distribute privileges to the community
- User requests access to resource with their cert.
  - If CAS decides they have sufficient privileges, they are sent a proxy certificate with an embedded policy allowing access to the resource

CAS server

- A CAS server is initiated for a community:
  - A community representative acquires a GSI credential to represent that community as a whole, and then runs a CAS server using that community identity.
  - Each resource provider verifies that the holder of the community credential represents that community and that the community’s policies are compatible with the resource provider’s own policies.
  - Once a trust relationship has been established, the resource provider then grants rights to the community identity, using normal local mechanisms (e.g. gridmap files and disk quotas, filesystem permissions, etc.)

CAS server

- Community representatives use the CAS to manage the community’s trust relationships
  - For example, to enroll users and resource providers into the community according to the community’s standards) and grant fine-grained access control to resources.
- The CAS server is also used to manage its own access control policies:
  - For example, community members who have the appropriate privileges may authorize additional community members to manage groups, grant permissions on some or all of the community’s resources, etc.

CAS server

- When a user wants to access resources served by the CAS, that user makes a request to the CAS server.
  - If the CAS server’s database indicates that the user has the appropriate privileges, the CAS issues the user a GSI restricted proxy credential with an embedded policy giving the user the right to perform the requested actions.
  - The user then uses the credentials from the CAS to connect to the resource with any normal Globus tool (e.g. GridFTP).
  - The resource then applies its local policy to determine the amount of access granted to the community, and further restricts that access based on the policy in the CAS credentials.
  - This serves to limit the user’s privileges to the intersection of those granted by the CAS to the user and those granted by the resource provider to the community.
VOMS – Virtual Organization Membership Service

- Part of European DataGrid project
- Supplies a database for storing authorisation data for users (and manipulation tools)
- Proxy credentials can be generated that contain VOMS AuthZ info as well as generic GSI info
- Very flexible –
  - Grid apps can use credentials and ignore VOMS data.
  - VOMS apps can use the AuthZ info.
  - Or both...

PERMIS

- Privilege and Role Management Infrastructure Standards Validation
  - A generic Java API for Authorization
    - Can protect many applications
  - Compliant with the GGF SAML API
  - PERMIS grants access to resources based on presented Attribute Certificates which are checked against the local policy...→

PERMIS Policy

- Subject Policy
  - Specifies the domain (as an LDAP subtree) of users who may be granted roles within the PMI

```xml
<SubjectPolicy>
  <SubjectDomainSpec ID="MyCompany">
    <Include LDAPDN="o=My Organisation,C=GB"/>
    <Exclude LDAPDN="ou=OtherCompany,o=MyOrganisation,C=GB"/>
  </SubjectDomainSpec>

  <SubjectDomainSpec ID="everyoneElse">
    <Include LDAPDN=""/>
  </SubjectDomainSpec>
</SubjectPolicy>
```

- Source Of Authority (SOA) policy
  - Lists the LDAP DNs of SOAs which are trusted to issue roles to the subjects specified above
  - First name listed is the LDAP DN of the policy creator (required), subsequent names are SOAs which are "cross-certified" by the policy creator
  - This name(s) will become the root issuer name(s) in a signed Attribute Certificate
    - Any trusted AC for this policy must have been signed by one of them

```xml
<SOAPolicy>
  <SOASpec ID="MyCompanyAdmin" LDAPDN="cn=Admin,o=MyOrganisation,C=GB"/>
</SOAPolicy>
```
PERMIS Policy

- Role Hierarchy Policy
  - Defines the role hierarchies supported by this policy
  - Specified as a "directed graph" of Superior-Subordinate attribute values
  - Each role named using an attribute type, attribute value pair (e.g. permisRole,Slave)

  <RoleHierarchyPolicy>
  <RoleSpec OID="1.2.345.0.1.321432.1.1.14" Type="permisRole">
    <SupRole Value="Boss"/>
    <SubRole Value="Slave"/>
  </RoleSpec>
  </RoleHierarchyPolicy>

PERMIS Policy

- Role Assignment Policy
  - Specifies which roles can be given to which subjects by which SOAs
  - Supports delegation and time constraints (not used here)

  <RoleAssignmentPolicy>
  <RoleAssignment ID="MyCompanyAdminAllocator">
    <SubjectDomain ID="MyCompany"/>
    <RoleList>
      <RoleType="permisRole" Value="Boss"/>
      <RoleType="permisRole" Value="Slave"/>
    </RoleList>
    <Delegate Depth="0"/>
    <SOA ID="MyCompanyAdmin"/>
    <Validity/>
  </RoleAssignment>
  </RoleAssignmentPolicy>

PERMIS Policy

- Repository Policy
  - Allows the PDP to search multiple LDAP directories
  - One of the policy statements that supports Dynamic Delegation (more later)

  <RepositoryPolicy>
  <LDAP="ldap://ldap.server.ac.uk"/>
  <LDAP="ldap://another.ldap.server.ac.uk"/>
  </RepositoryPolicy>

PERMIS Policy

- Target Policy
  - Specifies the target domains covered by this policy
  - Give the name of your Grid Service you want to protect here

  <TargetPolicy>
  <TargetDomainSpec ID="MyCompanyGridService">
    <Include URL="http://localhost:8080/ogsa/services/GridServices/core/first/MyCompanyGridService"/>
  </TargetDomainSpec>
  </TargetPolicy>
PERMIS Policy

- Action Policy
  - Defines the actions (operations on targets) supported by this policy
  - Let's say the MyCompany Grid Service operates the doors in the MyCompany building...
  - ‘Name’ is the specific command to perform the action

```xml
<ActionPolicy>
  <Action Args="MyCompanyGridService Name="lockMainDoor"/>
  <Action Args="MyCompanyGridService Name="unlockMainDoor"/>
  <Action Args="MyCompanyGridService Name="lockOfficeDoor"/>
  <Action Args="MyCompanyGridService Name="unlockOfficeDoor"/>
  <Action Args="MyCompanyGridService Name="getMainDoorStatus"/>
  <Action Args="MyCompanyGridService Name="getOfficeDoorStatus"/>
</ActionPolicy>
```

PERMIS Policy

- Target Access Policy
  - Specifies which roles are needed to access which targets for which actions (+ under what conditions)
  - Operates a Deny All Unless Specifically Granted rule
  - One must possess all roles within a target clause to gain access (may need multiple ACs to access)

```xml
<TargetAccessPolicy>
  <TargetAccess ID="public">
    <TargetList />
    <TargetList>
      <Target Domain ID="MyCompanyGridService" />
    </TargetList>
  </TargetAccess>
  <TargetAccess ID="slaves">
    <RoleList>
      <RoleType="permisRole" Value="slave" />
    </RoleList>
    <TargetList>
      <Target Actions="unlockOfficeDoor,lockOfficeDoor" />
      etc...
    </TargetList>
  </TargetAccess>
</TargetAccessPolicy>
```

Another Problem...

- The Source of Authority signs ACs
  - He holds the SoA key pair
  - But he may want to delegate the ability to sign ACs to someone else
    - And they may wish to delegate the ability down again...
    - Remember these are longer lasting certificates than proxies
    - And a CA doesn’t want you signing certs on their behalf!
  - Each of these people will require a key pair within the PMI to sign ACs
    - Not such a big deal??
  - What happens if one of the certificates in this chain needs to be revoked?
    - The chain above this certificate becomes invalid
    - When a service tries to use the certificate, the trust chain checker finds that one certificate in the chain can’t be verified — DENIED!!
Delegation Issuing Service

- DIS creates a special user (called DIS) who holds a key pair and will be responsible for signing all the ACs in the PMI
  - Umm.. Aren’t we doing this already!?
  - No, this DIS user can be accessed by anyone in the PMI who is authorised by the SoA
    - The SoA delegates ability to assign attributes to subordinate USERS by giving them ACs with a DELEGATION STATEMENT
    - SoA gets ability from the Delegation part of the Policy
    - The USER logs into the DIS and assigns roles based on the abilities they have been delegated from their AC
    - BUT when they sign the certificate, it is actually the DIS that signs it
      - This means that any certificates that get revoked will not affect any others

Delegation Issuing Service

- There are two more tools being developed by PERMIS which aim to provide a dynamic way to establish VOs
  - Both may be implemented NOW through policy manipulation
    - But we want to get away from that as the policy should be set in stone, VOs should connect not dictate policies
  - Role Mapping
    - The equating of roles at different institutions and any inheritance
  - Cross-Certification
    - The recognition of an external Source of Authority

Authorisation in the Future

- PERMIS is providing a powerful set of tools to allow complex ACs to be created and verified with dynamic VO capabilities
  - VOMS ACs are directly compatible with Globus Toolkit
  - A powerful Grid Authorisation infrastructure will feature a composite of these functions
    - Is this what we asked for in slide 6 (“What We Need Is?”) ???
      - Hopefully some of you may be asked to give the answer!
    - But the future is Privilege Management…
Summary

• Authorisation has many solutions which may be complementary

• The future Grid will have to utilise Privilege Management if it is to scale

• Push to make access to the Grid as easy as possible (use existing passwords/Shibboleth)