Fine-Grain Authorization for Resource Management in the Grid Environment

Naiwen Lin
Introduction

- Increasing trend to allow the use of both hardware and software resources based on VO credentials.

- Sharing VO-owned application services requires VO-wide mechanisms for managing both these services and the VO’s resources usage rights.
Introduction

- In this paper, we present an architecture relying on VO credential for service and resource management, allowing us to specify and enforce VO-wide service and resource usage policies.

- Propose changes and extensions to the current Globus Toolkit’s (GT2) resource management mechanism.
Scenarios and Requirements

- **VOs** are interested in setting policies in terms of
  - who
  - what
  - how

- **Combining policies from different sources**
  - Policies from the resource owner
  - Policies from the VO
Scenarios and Requirements (2)

- **Fine-grain control of how resources are used**
  - one group has the role of developing, installing, and debugging the application services.
  - Need a large degree of freedom in the types of apps., but should only consume small amounts of traditional computing resources (CPU, disk, bandwidth).

- **Another group runs analysis**
  - Consume large amounts of resources
  - Use applications services approved by the VO
Scenarios and Requirements (3)

- VO-wide management of jobs and resource allocation
  - Currently, the only users who are allowed to manage a job are those who instantiated it and the administrators.
  - VO wants to give a groups of its members the ability to manage any jobs
GRAM System

- The current Globus Toolkit GRAM (Grid Resource Acquisition and Management)
- Has two components
  - Gatekeeper
    - Responsible for creating a Grid service requested by the user
  - JMI (Job Manager Instance)
    - Provides resource management and job control
Gatekeeper

- Responsible for authenticating and authorizing a Grid user
- Authentication is based on the user’s grid credential and an access control list in a file `grid-mapfile`
- The file is used to map the user’s Grid identity to a local account.
- Gatekeeper starts up a JMI
Job Manager Instance

- Parse the user’s job startup request
- Interface with resource’s job control system
- Monitor job progress
- Handle job management requests from the user
- Authorization policy – must be the same user who initiated the job
GRAM shortcomings

- Authorization of Grid service and user job startup is coarse-grained
- Authorization on job management is coarse-grained (must be the same user)
- JMI runs under local user credential and useless for enforcing fine-grained policy
- Local enforcement depends on the rights attached to the user’s account, not associated with the request and Grid credential
- A local account must exist for a user
Proposed Authorization and Enforcement Extensions to GRAM

- Authorization System Extensions
- Policy Language
- Policy Enforcement
  - Implementing enforcement in GRAM
  - Dynamic Accounts and Sandboxing
Authorization System Extensions

- Capable of evaluating complex fine-grain policies coming from the resource provider and the VO
- Currently working with two systems
  - Akenti
  - Community Authorization Service (CAS)
Policy Language

- In GRAM, the job description is formulated in terms of attributes specified by the Resource Specification Language (RSL)

- RSL consists of attribute value pairs specifying executable description (name, location) and resource requirements (number of CPUs, memory)

- Currently designing a policy language
  - Specify the job description in terms of RSL and concepts such as actions, job ownership, and jobtags
Jobtag

- Jobtag allows us to make policy about those jobs
- Extend RSL to accept a jobtag as a parameter
- At present, jobtags are defined statically by a policy administrator
- Envision an approach in which the users will define them dynamically.
Policy Enforcement

- Implementing enforcement in GRAM
  - Create a gateway controlling all external access to a resource

- Dynamic Accounts and Sandboxing
Dynamic Accounts and Sandboxing

- A sandbox is an environment that imposes restrictions on resource usage.
- Sandboxing is largely complementary to the gateway approach.
- Dynamic accounts are accounts that are created and configured on the fly.
Summary

- Describe a work in progress aiming to provide mechanisms for VO-wide authorization and enforcement.

- Design a system to support fine-grain authorization on job startup and management.
A Community Authorization Service for Group Collaboration

Naiwen Lin
Introduction

- With current technologies, each change in personnel at participating institutions requires that the project leader contact the resource owner to create an account for each new team member.
- As project policies change, the project leader will have to adjust allocation, rights and priorities.
Challenges of Policy Enforcement

- **Scalability**
  - Adding or removing participants, changing community policy
  - Administration overheads should be bounded

- **Flexibility and expressibility**
  - Policies will vary over time
  - Enforcement introduces difficult bookkeeping issues

- **Policy Hierarchy**
  - Nested policies must be consistent
In this paper

- Design and implement a Community Authorization Service (CAS)
- Keep track of its membership and fine-grained access control policies
- CAS builds on public key authentication and delegation mechanism provided by GSI
Grid Service Infrastructure

- GSI is a set of libraries and tools that allows users and applications to access resources securely.
- Proxy credential
  - Temporary credentials
- Delegation
  - Delegate a proxy credential to a process on a remote host
- Authorization
  - Translating users’ GSI identity to a local identity
  - Local identity can be used to enforce local policy decisions
Community Authorization

- Expressing policies in terms of direct trust relationship between producers and consumers has the problems of scalability, flexibility, expressibility, and lack of policy hierarchy.

- So we introduce a trusted third party, a CAS server responsible for managing the policies.
CAS

- The CAS server contains entries for CAs, users, servers and resources.
- Specify
  - Who (user or group)
  - Which (resource or resource group)
  - What permission
CAS Server

What rights does the community grant to this user?

CAS request authenticated with user credential

CAS reply with Capability

User

CAS-Maintained Community policy database

Figure 1: In order to gain access to a CAS-managed community resource, a user must first acquire a capability from the CAS server.
Figure 2: A CAS capability is used to authenticate to the resource. This action can be repeated using the same capability until it expires.
CAS

- The structure addresses the scalability problem by reducing the necessary trust relationships from CxP to C+P

- But CAS server itself is a potential bottleneck and single point of failure
It’s inappropriate for the CAS server to delegate all of its authority to a user.

We extended the GSI delegation feature to support rich restriction policies.

A proxy carrying such restriction policy is called a restricted proxy.

Delegate the user a restricted proxy credential and limit what the user can do.
Security Considerations

- Restricted Proxy Certificates
- Compromised CAS server
- Revocation Mechanism
- Compromised Resource Server
Future Directions

- Accounting
- Delegation tracing
- Replication of CAS server
Summary

- Describe CAS server to solve three critical authorization problems: **scalability**, **flexibility**, and **expressibility**.
- Introduce trusted third party administered by a VO
An Online Credential Repository for the Grid: MyProxy

Naiwen Lin
Introduction

- Grid Portals are a common approach to providing user interfaces.
- Unfortunately, standard Web security protocol do not support the needs of Grid Portals.
- This paper describes MyProxy, designed and developed to bridge this incompatibility between Web and Grid security.
Grid Portal Requirements

- Users must be able to use any standard web browser to access the Grid portals.
- Users must be able to use a web browser from locations where their credentials would not normally be available to them.
- Users must be able to do anything through a Grid portal that their credentials would entitle them to do.
Existing Constraints

- Grid credentials are typically stored as files on a file system
  - A user is unable to access their credentials when away from their primary system
- Not all applications are Grid-enabled
  - Although the users can authenticate to the portal, they cannot delegate authority to the portal to act on their behalf
Goals for MyProxy System

- Allow users to access credentials from anywhere on the Grid
- Allow them to delegate credentials to resources
- Remove any credential from the portal except when they are actually needed
- Scalable
- More control of credential
MyProxy Credential Repository System

- Two basic steps
  - Delegation of a proxy credential to the repository
  - Retrieving the credential from the repository
Figure 1. MyProxy-init process
Figure 2. MyProxy retrieval process
Figure 3. User connecting to portal and portal interaction with MyProxy repository.
Repository Issues

- If the repository is compromised, an attacker could potentially gain access to these credentials
  - The repository should be run on a tightly secured host
  - Encrypt the credentials with the pass phrase
Portal Issues

- Use HTTPS to transmit the name and pass phrase
- The credentials maintained by portal are unencrypted
- Map the credentials to the user’s web session
  - Cookies or rewrite URL
Future Directions

- Managing permanent user credentials
- Managing multiple user credentials
- Alternate authentication mechanism
- Standardize client-server protocol
- Restricted Proxy credentials
- Support for Condor-G
Summary

- Describe an online credential repository: MyProxy
- Enable Grid Portals to access GSI-protected resources