A SCALABLE GRID USER MANAGEMENT SYSTEM FOR LARGE VIRTUAL ORGANIZATION

G. Carcassi, T. Carter, Z. Liu, G. Smith, J. Smith, J. Spiletic, T. Wlodek, D. Yu, X. Zhao,
Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

We describe our work on GUMS, a site tool for Resource Authorization (AuthZ) and Grid User Identity Mapping. We will first define the scope of the work, and describe the general direction we are taking. We will describe the current functionalities provided to BNL, such as the ability to have a flexible site policy controlled by a single XML file and the ability to integrate with site databases. We will then describe the current work being done for OSG, which includes the use of account pool, a GT3/4 based service and role based authorization using VOMS extended proxy credentials.

OVERVIEW

GUMS (Grid User Management System) is a site tool for resource Authorization that addresses one function: mapping grid certificates to local identities (i.e. UNIX account).

Grid Identity Mapping Service

A job comes to a site with a GRID credential (the proxy certificate). The site resources might not use GRID credentials natively, and will use some different mechanism to identify users, such as UNIX accounts, Kerberos principals, and the like. The gatekeeper will need to map the GRID credential to the site credential. GUMS is a service that provide this type of mapping: tells you which site user the GRID user should be using. Notice that it doesn't authenticate for you: it doesn't 'su', it doesn't retrieve Kerberos credentials. It just tells the gatekeeper which site credentials should get. The gatekeeper is still in charge of enforcing the site mapping established by GUMS.

Past, Present and Future

GUMS was first designed by Rich Baker and Dantong Yu at BNL in the first half of 2003. A first implementation was provided by Tomasz Wlodek and Dantong Yu. Gabriele Carcassi took over the project in March 2004 and brought the system into full production at BNL in May 2004. Between June and July the code was consolidated to allow the business logic to be called either from command line or a web application, which allows a GT3/4 service implementation.

Current work is going toward a web application that would provide both a web interface for the administrator and a web service that implement the OGSA AuthZ interface. This is done within the Privilege Project, a joint project between USCMS and USATLAS.

CURRENT FEATURES

GUMS is being used in production at BNL, proving it's a viable tool to managed account mapping and resource authorization.

We here describe the main functionalities GUMS provide to our facility.

Policy based mapping

GUMS allows to have a single, site-wide policy for user mapping. This means that a single XML configuration file controls how different gatekeepers will map Grid credentials to local accounts. The resource administrator is able to assign different mapping policies to different group of users. He can also define groups of hosts on which these mappings will be used.

For details and XML syntax you can refer to the online manual [1]. Here we describe a couple of examples, inspired by our production system at BNL:

- First, we define the "usatlas" mapping at our site. It will consist of all the members of the usatlas group within the ATLAS LDAP VO. We will map these users according to a special mapping table provided by a database, which allows to handle special cases. If no entry is found, the mapping will use an account from a generic pool of pre-created accounts; the account information is stored in another database. If the pool of accounts is exhausted, we will map the user to a generic account "usatlas".
- We can go on and define other mappings, for example the "star" mapping. It will consist of all the users within the STAR VOMS server. We will map these according to a special mapping table we have on a database, which allows to handle special cases. If no entry is found, we will try to map to a local account by using name and surname from the certificate. If no precise match was found (i.e. no exact name/surname, more than one account returned), no mapping is done, and the user won't have access to the resource.
- Now we define a first group of hosts with a wildcard: "star*.mysite.gov". On these hosts we will use the "star" mapping, and nothing else. We also define the "atlas*.mysite.gov". On these hosts we will the following lists of mappings: atlasProd,

usatlas, atlas, cms and ligo (we assume all the other mappings were defined). GUMS will look for the mapping for a user in the first group; if none is found it will proceed to the second group until either the list is exhausted or an account was returned.

GUMS allows to combine group definitions, mapping policies and host groups at runtime by changing the XML configuration file. It also allows to create your own policies or group definitions from scratch.

Open architecture for site integration

Most components, like mapping policies or groups, are identified by Java interfaces whose implementations are chosen at runtime in the configuration file. GUMS was designed so that any resource admininistrator can wrap some code, with very little knowledge of GUMS itself, to integrate into their systems. In fact, it is crucial for GUMS to be able to integrate with existing site schemes since it's meant to be a site tool.

For example, a site might have a policy of storing all the access information in a specific database, or LDAP. GUMS must allow, and it does allow, storing all its information on such a system.

The manual [1] will provide instructions on how to do that, as well as examples taken from the actual code. In fact, most of GUMS components are implementations of those interfaces.

Components

The components being used in production include the following.

Persistence layers:

A persistence layer is responsible to read/write all the information that GUMS uses. To use a site database and information systems, one only need to create his own layer.

- MySQL. We currently provide a simple persistence mechanism to MySQL. This can be used in production. A site probably wants to integrate with their systems.

Groups:

Groups are used to define a series of users that are going to be mapped according to a specific policy

- VOMS and LDAP groups. Groups of users can be specified in GUMS by referring to groups defined in VOMS or in the lcg LDAP VOs.
- Manual groups. A group of users can be manually defined by a table provided by the persistence layer.

Mapping Policies:

Mapping policies define how to map a grid certificate to a local account

- Group account: all the users are mapped to the same account
- NIS mapping: the certificate name is compared to the gecos field to find a match for the user account. This is meant to be a best effort policy, and its output should be supervised by the resource administrator.
- Account pool: the certificate is mapped to a generic account taken from a pool. We'll talk more about this in the following section.
- Manual mapping: the mapping is done according to a table provided by the persistence layer.
- Composite mapping policy: combines a list of policies with a fallback mechanism; if the first policy doesn't return a mapping for the user, the second is used. It returns the user selected by the first matching policy.

CURRENT WORK

Work on GUMS is continuing, with an emphasis on the following areas:

- Account pool
- Grid Service implementation
- Role based authorization

Account pool

The requirement to allow auditing on grid jobs can be solved by the use of a pool of pre-authorized accounts. When a user comes in with a job, he is assigned an account from the pool. This way all process and files created by the job can be audited through standard tools.

We won't address the problem of account recycling at first for the following reasons:

- To recycle an account the system must be sure that all the processes and files created by all the jobs submitted by the user are eliminated. Currently, this is a difficult problem.
- The number of accounts that can be created on a Unix system is sufficiently large that account recycling is not critical.

One of the problems we do have to face is to guarantee different file permissions within and outside VO membership. We envision solving this problem using Unix groups.

When an account is assigned to a user, depending on the VO membership, a group is also assigned. This means integrating GUMS with the site systems, such as NIS and LDAP. We will provide all the necessary hooks for a site to perform such integration, and create an implementation for BNL which can be used as a prototype.

The account pooling managed by GUMS is already in production in a test environment at BNL, which will allow members of Grid3/OSG to test their applications.

Grid service implementation

GUMS business logic is implemented in a way that allows it to be called from different presentation layers, such as command line and web. The core library consists on a series of methods designed to be called from any environment, and the web or command line module is built on top of that. Therefore, even though GUMS started as a command line tool, it is evolving into a web service application with a web service interface. We are

developing on top of the Globus Toolkit v3.x, and we expect to port it to v4.x once a sufficiently stable release becomes available.

Within the Privilege Project, a joint project between USATLAS and USCMS, we are developing the protocol that will be used between the gatekeeper and the authorization service. GUMS is an implementation of this protocol. The interface definition is lead by Markus Lorch [2] [3] [4] and is being discussed within GGF [5].

Work is also being done to investigate the scalability and performance issues of developing a Grid service for the Globus Toolkit 3.x.

Role based authorization

Another functionality being pursued as part of the Privilege Project is role based authentication. This means using the extended proxy credentials created by a VOMS server to decide which account will be used at local site.

The user, though the 'voms-proxy-init' command, will ask his VO to create a proxy specifying a role, a group and/or a capability. The VOMS server will create this type of proxy, sign it, and give it back to the user. Once a job is submitted, the gatekeeper forward the information inside the proxy to GUMS, which can use that information to return a different local user depending on vo, group, role, capability or any combination of them.

For example, a VO could have an "application manager" role, which allows the user to install/remove VO application from the site. GUMS would map users coming in with that role to a different user that has special privileges.

This work requires the Grid service implementation. In fact, the protocol between the gatekeeper and the authorization service is being designed with these requirements in mind.

CONCLUSION

We discussed GUMS, a resource authorization tool to perform grid user identity mapping. GUMS is currently being used in production at Brookhaven National Laboratory, and the development is

If you need a tool to manage grid user identity mapping within your facility, which allows integration with your site information services, GUMS might be the tool for you.

REFERENCES

[1] http://grid.racf.bnl.gov/GUMS.

[2] Markus Lorch, Dennis Kafura, "The PRIMA Grid Authorization System", submitted to the Int. Journal of Grid Computing, July 2004 http://zuni.cs.vt.edu/publications/jogc-prima-july2004.pdf.

[3] Markus Lorch, David Adams, Dennis Kafura, Madhu Koneni, Anand Rathi, Sumit Shah, "The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments", Proc. 4th Int. Workshop on Grid Computing.

[4] Grid 2003, 17 November 2003 in Phoenix, AR, USA. http://zuni.cs.vt.edu/publications/PRIMA-2003.pdf.

[5] OGSA Authorization Working Group: https://forge.gridforum.org/projects/ogsa-authz/document/Use_of_SAML_for_OGSA_Authorization_with_Obligations/en/1

[6] Alfieri et al. "VOMS: an Authorization System for Virtual Organizations" 1st European Across Grids Conference, Santiago de Compostela, Feb. 13-14, 2003

[7] V. Sehkri, I. Mandrichenko, D. Skow, "Site Authorization Service (SAZ)", Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, CA, USA, March 2003, available from http://arxiv.org/pdf/cs.DC/0306100