University of Kent

Integrating VOMS and PERMIS for Superior Secure Grid Management (VPman)

Deliverable 1.1
Requirements and information gathering

Version <1.0>

Revision History

Date	Version	Description	Author
5/April/2007	0.1	Initial draft	Bassem Nasser
5 April	0.2	revisions	David Chadwick
20 April	0.3	updates	Bassem Nasser
18 June	0.4	updates	David Chadwick
19 June	0.5	updates	Richard Sinnott
20 June	0.6	Updates	David Martin
23 June	0.7	Updates	Bassem Nasser
11 July	1.0	Final update	David Chadwick

Table of Contents

VPman background and literature overview

1. Introduction

1.1 Purpose

The Virtual Organization Membership Service (VOMS) was originally developed in the framework of EDG and DataTAG [39] collaborations to solve the problems of granting users authorization to access the resources at VO level, providing support for group membership and roles. It is now maintained within the EU project Enabling Grid for E-SciencE (EGEE). In addition to finer grain security, VOMS benefits EGEE by reducing the maintenance load on sites in respect of authorized user lists and also in the future by providing a handle for allocating Quality of Service (QoS) to jobs.

PERMIS is an authorisation system developed by the ISSR-Group at the University of Kent. PERMIS makes the access control decisions based on a role-based access control policy. Moreover, PERMIS has many features such as enabling dynamic delegation of authority, obligation policies and separation of duty enforcement.

Both VOMS [1] and PERMIS [2] provide security management infrastructures for Grids but are predominantly used by different groups of Grid users. Each has its strengths and weaknesses and their combination would be a powerful solution to Grid security management as described below. Currently however they are not integrated and the benefits of a combined system cannot be enjoyed. This project proposes to address this directly. Specifically the objectives of this project are to:

- integrate VOMS and PERMIS, more specifically the VOMS user management and attribute assignment function with the PERMIS policy based authorisation decision function;
- ensure they seamlessly inter-work with latest Grid technologies including Globus toolkit version 4 (GT4)[3], the Open Middleware Infrastructure Institute UK (OMII-UK) [4] software release and Shibboleth [5];
- validate the results in several representative major pilot applications run by the National e-Science Centre (NeSC) at the University of Glasgow;
- evaluate the combined software from user, administrator and Grid developer perspectives;
- integrate the combined infrastructure with the National Grid Service (NGS) at SciTech (formerly CCLRC);
- distribute the integrated software as open source code as part of either Globus Toolkit, or the OMII-UK Repository, or the US-NMI, or a combination of them.

This document is part of WP1. It provides a literature overview of the background to the integration work to de done in this project.

1.2 Definitions, Acronyms, and Abbreviations

AA	Attribute Authority
AC	Attribute Certificate
ACL	Access Control List
CA	Certification Authority
DN	X.500/LDAP Distinguished Name
GA	VOMS Generic Attribute
GACL	Grid ACL
GT	Globus Toolkit
Gums	Grid User Management System

IdP	Identity Provider
LCAS	Local Centre Authorization Service
LCMAPS	Local Credential Mapping Service
LDAP	Lightweight Directory Access Protocol
OMII-UK	Open Middleware Infrastructure Institute UK
PDP	Policy Decision Point
PERMIS	Policy-basEd pRivilege Management InfraStructure
PIP	Policy Information Point
SP	Service Provider
VO	Virtual Organisation
VOMRS	Virtual Organisation Management Registration Service
VOMS	Virtual Organisation Membership Service

1.3 Overview

Managing grids from a security perspective comprises two main functions: the privilege assignment function in which users are assigned to roles and the authorisation decision function in which policies are set for which roles should have access to which grid resources. These functions typically take place in different systems at different locations. These functions are carried out by the Identity Provider (IdP) and Service Provider (SP) in Shibboleth terminology, and by the VO Manager and grid service provider in grid terminology.

More generally, privileges are assigned to users as a mixture of attributes and roles, by one or more attribute authorities (AAs) or IdPs. Attributes (such as login id and department) are assigned by a user's home institution; virtual organisation (VO) roles are assigned by a VO management authority, and potentially by professional memberships and learned societies such as IEEE and ACM. These attributes are then transferred to the grid SP, where the authorisation decision function is carried out based on the policy set by the resource's owner. If a user's grid jobs are sent to multiple resources at multiple sites, then the authorisation decision function may take place several times at several different resource sites using different policies in each case. The Virtual Organisation Management Service (VOMS) [1] provides a well utilised privilege assignment function which is carried out by the VO manager. It is the chosen VO management function of Grid projects such as EGEE, and it is planned to integrate it into the National Grid Service (NGS) at SciTech (formerly CCLRC).

However, its authorization decision function is intentionally missing by design (it relies on LCMAPS and LCAS, see later). PERMIS on the other hand provides a feature rich, modular authorisation decision function, with a user friendly policy management interface; it is already integrated into Shibboleth and is currently being integrated into the OMII-UK software environment by the London E-Science Centre (LeSC). But it has a less well developed privilege assignment function.

This project proposes to integrate the privilege assignment function of VOMS with the authorisation decision function of PERMIS, so that the management of grids becomes easier, whilst simultaneously allowing finer grained more feature rich authorisation infrastructures to be designed and built. We expect the combination of these technologies to have a significant impact across the UK and international e-Science communities.

2. Background

2.1 Background to PERMIS

The main strength of PERMIS, and the primary focus in its development, has been its modular

construction [16] and compliance with standards. It supports X.509 attribute certificates [7], policies in XML, Shibboleth attribute assertions, the OGSA Authz protocol [17], LDAP repositories and has an optional XACML interface [15].

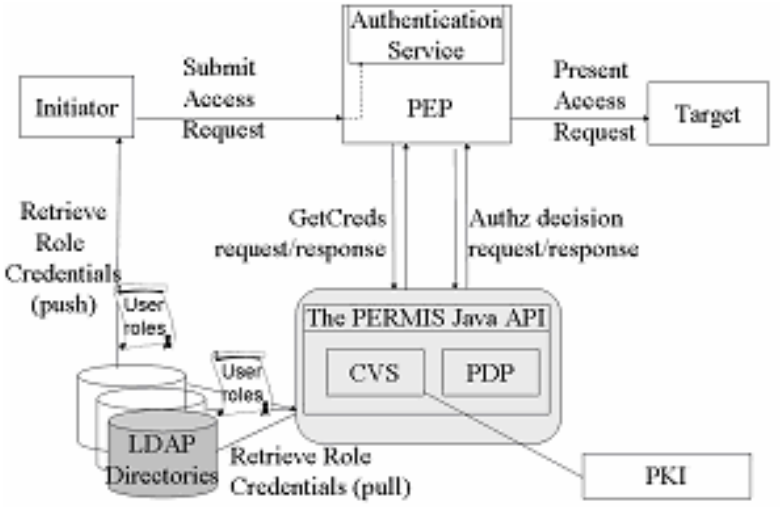

Figure 1. PERMIS infrastructure

PERMIS's authorisation decision function comprises two components: a credential validation service (CVS), and a policy decision point (PDP). Both components are policy driven. CVS policies are of the form "this authority is trusted to assign these attributes to this group of users, and delegation to depth n is allowed". PDP policies are of the form "users with this set of attributes are allowed this type of access to this resource, providing that the following conditions are met". The PERMIS PDP provides similar functionality to the XACML PDP (although both have features the other does not support). XACML has no equivalent functionality to the CVS [18]. Some of the more notable features of PERMIS include:

2.1.1 Security policy

Authorization policies can be created via a user friendly GUI [19] and a new Policy Wizard. This policy is written in XML and may be inserted in an attribute certificate to ensure integrity. Certificates may be stored on LDAP or WebDAV servers [37]. As indicated above, the policy is role-based indicating that "users having certain *roles* are authorized to do *actions* on *resource*(s)".
Arbitrary conditions can be set on policy rules such as the time of day and the user's request e.g. only grant access if requested storage is less than 30 GB or the local time is after 8pm. Moreover, Obligation policies are also supported by PERMIS. This feature allows resource owners to insert obligations into their access control policies, requiring the application to enforce obligations on the user's access request prior to or after granting access to the resource e.g. run this job under username xdp12, or debit the user's account with 24 units.

2.1.2 Dynamic delegation

Users may be allowed to dynamically delegate their privilege attributes to other users for a specific period of time e.g. a researcher may delegate the "guest of project X" attribute to a colleague to allow the latter to run a series of experiments for one week. A Delegation Issuing Service (DIS) [11] is implemented supporting these kinds of scenario. The DIS delegates attribute certificates to delegatees on behalf of users (delegators) whilst simultaneously enforcing the role allocation (or delegation) policy of the delegating site. Note that neither SAML [20] nor XACML[21] are able to

support this feature since issuers and subjects are specified differently, hence chaining credentials together cannot be supported.

2.1.3 Separation of duty

Separation of duties (SoD) is a key security requirement for many business and information systems. PERMIS goes beyond traditional RBAC session based Separation of Duty (SoD). It proposes a **multi-session** SoD (MSoD) policies for business processes which include multiple tasks enacted by multiple users over many user access control sessions. Users can be forbidden from performing multiple tasks or holding conflicting roles that would allow mistakes or fraud to occur, e.g. it can require that two different scientists validate a set of results before they are released to the public domain. MSoD policies are expressed via multi-session mutually exclusive roles (MMER) and multi-session mutually exclusive privileges (MMEP). PERMIS enforces this policy since its PDP is rendered stateful using history-based authorization decisions (an authorization decision depends on past ones). The publicly available XACML PDP cannot do this as it is a stateless PDP.

2.1.4 PERMIS user's attributes

PERMIS has concentrated much less on the way that attributes are assigned to users. Indeed the PERMIS decision engine does not care how this is done, and assumes it can be done in many different ways. Consequently, PERMIS supports a set of policy rules (in the CVS) to constrain which attribute assignments are trusted/allowed to take place. PERMIS can let other software assign attributes, for example, organisations may assign attributes to users using SIGNET and GROUPER [22], as is supported in the PERMIS-Shibboleth integration. PERMIS also provides its own tools to make attribute assignments (as digitally signed X.509 attribute certificates (ACs)) and these are stored in LDAP directories. The Delegation Issuing Service of PERMIS [11] uses these digitally signed ACs to allow users to dynamically assign (a subset of their) attributes to other users. In all cases the PERMIS CVS uses its policy rules to validate the attribute assignments, regardless of which software produced them.

2.2 Background to VOMS

To understand the environment in which VOMS is meant to operate, it is necessary to introduce Globus-based Grid Security Infrastructure (GSI) an enabling infrastructure to use VOMS credentials.

2.2.1 Globus Security Infrastructure GSI

The Grid has requirements for long-running collaborations that federate the use of many distributed resources. Consequently, it is necessary for (remote) programs to operate on a user's behalf without the user being present. The user must be prepared to delegate authority to a program and the program must be able to authenticate itself as an entity having that authority. Globus GSI specialises in this aspect of delegation. It has been set up to accept proxy-certificates as authenticating the user who issued them. So, a user wishing to delegate authority, acts like a CA and issues a public key certificate signed by themselves using their own public key certificate. A proxy-certificate is short lived (a few hours, typically) but gives the program possessing it access to resources with the authority of the user that signed it. Recursive delegation is of course possible. Note that proxy certificate delegation is fundamentally different to PERMIS delegation of authority. With proxy certificate delegation a user delegates a public key certificate effectively to himself (actually to his grid job) so that it can authenticate as the user's proxy. With PERMIS delegation, one user delegates privileges to another user, in the form of attribute certificates, and the delegate then authenticates as himself using his own public key certificate (or other mechanism). PERMIS delegation does not have to be short lived. It can be for as long as the delegator requires.

2.2.2 VOMS server

VOMS, in its own words is “basically a simple account database, which serves information (VOMs credentials) to insert in a user proxy certificate. The VO manager can administrate VOMS remotely using command line tools or a web interface [6]. Even though it is only a simple account database, nevertheless the account management is well developed and has the concepts of groups, subgroups, roles and capabilities (although capabilities are now deprecated). Groups may contain subgroups nested to any depth, with the most superior group being the VO. A member of any subgroup is also a member of all the superior encapsulating groups up to that of the VO. Roles are assigned to VO members within a group context. Roles signify the roles a user has within a group context. Capabilities used to signify permissions to perform certain tasks within a group context, but capabilities are now deprecated and therefore no longer need to be considered by this project. When a user runs a grid job, all their group memberships are automatically included in their credentials that accompany the request, but the user can choose which roles to include in these credentials. Roles are encoded as free format strings, and so in principle can contain anything that the authorization decision function will understand.

The user credentials are actually encoded as X.509 attribute certificates [12, 13] containing fully qualified attribute names (FQAN) in the following format:

/VO[/group[/subgroup(s)]][/Role=role][/Capability=NULL]

VOMS is attempting to rectify some of its deficiencies with the G-PBOX work [14], which is producing a set of tools for policy management that will allow XACML policies and VOMS attributes to be used for authorising Grid jobs. The interface between the Grid application and the XACML PDP is an XACML request/response context, but it is not known at this time what protocol will be used to carry the XACML contexts. In parallel with (but separate from) the G-PBOX work, the OGF OGSA_AuthZ group is defining an XACML profile to be used for carrying the XACML request/response context between a PEP and a PDP for grids [15]. PERMIS is migrating to this protocol. The VOMS architects have been invited to join in this work so that a common standard for this interaction can be defined. It is proposed to use this OGF profile in this work.

2.2.3 VOMS Architecture

VOMS allocates unsigned plain attributes to users and stores these in its VOMS database. VOMS requires the VO Manager to delegate attributes to users. ACs (possibly with multiple attributes), signed by the VOMS server, are then created on demand when a user initiates a grid job. The client software embeds the user’s AC in a proprietary certificate extension [38]. Since the embedded AC is signed by a VOMS server, a VOMS enabled service can parse and verify this extra certificate and treat the data therein as extra information about the client to use in an authorization decision. The components of the system [24] are:

- A VOMS server, typically a VO-specific service, contains information about users who are identified by their DNs.
- The VOMS server, when requested, will digitally sign an assertion stating that a particular DN has some particular attributes
- A client may embed this in its own proxy certificate to "push" it to the service when accessing resources.
- The service, trusting a particular set of VOMS servers for any attribute information, can use the attributes to make authorization decisions

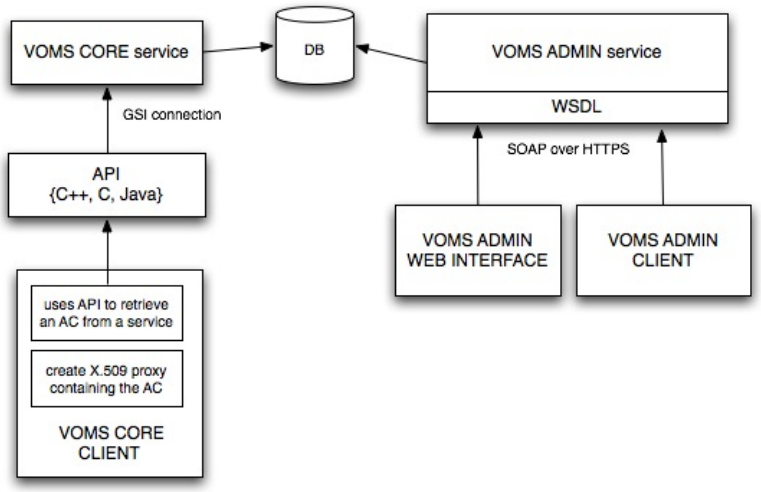

Figure 2. VOMS system

In a slightly more low-level view, it may be noticed that the components described are not the only ones that are present. Chief among the previously omitted ones, there is an administration interface, used to edit the database contents. This interface can be accessed through a browser or through a perl Command Line Interface. Also supported are a set of APIs, in C, C++ and Java, which are capable both of interpreting the attributes certificates when they are encountered and of contacting the server to create a new AC.

Technically speaking, the VOMS client (voms-proxy-init) enables VOMS credentials to be inserted into client proxy certificates. Thus after VOMS has been setup, users are supposed to replace the use of grid-proxy-init with voms-proxy-init to generate their proxy certificates, which are backward-compatible with the ones generated by the grid-proxy-init command and in addition contain extra information about the user and the VOs he belongs to. The user interface also offers some ancillary utilities such as voms-proxy-info, voms-proxy-destroy, voms-proxy-list.

VOMS supports multiple VO authorities allowing users to include multiple attribute certificates from multiple VOMS servers in their proxy certificates, providing a user has the same distinguished name at each server.

Finally, for backwards compatibility reasons, there is a mkgridmap service capable of creating gridmap-files (see below) from a VOMS server instead of from an LDAP server. However, as more and more services shy away from gridmapfiles and move to VOMS-based authorization, this feature will be removed.

2.2.4 VOMS user management

Members of a Virtual Organization can be organized into groups [28]. These groups can then be directly represented as voms groups. Groups are organized in a hierarchical tree, where each group may have zero, one or more subgroups, with no limitation on tree depth. The root of the tree is fixed, and is the VO itself.

A group name contains the representation of the path leading to it from the root. For example, if a user were member of the subgroup "*CS*" of the group "*Kent*" in the VO called "*VPMan*", the group name as represented by VOMS will be */VPMan/Kent/CS*.

- **Groups**

VOMS *Groups* are used to represent special communities within a VO, by putting them into *sub-groups*. There are no effective limits on the length of a group name, except those enforced by the underlying DB in which the name is stored. However, only alphanumeric characters plus '-', '_'and '.' are allowed in group names.

Membership in a subgroup requires membership in a parent group. From this it should be evident that all users must at least be members of the root group i.e. members of the VO.

- **Roles**

Not all members of a group are necessarily equal, but some may occasionally have some special rights that, while not always needed, may indeed be necessary for the execution of special tasks. A simple example of this is the software manager or sgm role that supports privileges necessary for installing software on the grid nodes of the VO. This need is represented by VOMS Roles. A VOMS Role is always associated to a specific group, e.g., holding the role sgm in the */VPMan/Kent* group is a different thing to holding the same role in the */VPMan* group.

Q. Which provides most privileges? Being the sgm for the VO or for a subgroup in the VO?

In order to answer this question, recall that the process of populating the VOMS database is first to create a group, then add the user to the group, then create a role. The role at this level is an independent entity (not related to the VO level or any specific group level). Finally, the role can be assigned to a user in the context of a group. Of course this group may be the VO or any subgroup. So from VOMS' perspective, these are just relations among a role, group, and user. Talking about privileges here is a bit tricky since it is the authorization policy that specifies the privileges and decides whether to:

- differentiate between the roles at the different levels or not, and
- give some role at a certain level different privileges from the same role at another level, or not.

So these semantics are out of scope from VOMS' point of view.

Note that when a user is removed from a group, all his roles within this specific group are revoked as well.

Restrictions on role names are the same as restrictions on group names. Furthermore, Role names starting with VOMS are reserved for use by VOMS itself.

The special name NULL is used to indicate that a user may not hold any specific roles or capabilities. On the face of it, this seems to be rather pointless. Why would the VOMS server need to tell the grid resource that the user does not have a roles, when it can simply be omitted?, This is due to a current limitation in VOMS implementations. VOMS defines two forms of FQAN: long forms (where Role=xxx Cap=NULL is mandatory) and short forms (where the Role and Cap may be omitted). For now VOMS implementations only work with the long format so the system has to use the special name NULL to indicate no roles and capabilities. In the future the short format may be adopted.

- **Generic Attributes (GA)**

It should be noted that not all the characteristics of a user can be represented by a combination of groups and roles. For example, consider the need for registering the guarantor of a user. For these cases, generic attributes are used. They consist of the triple (*name, value, qualifier*), with the qualifier being optional. As an example, the guarantor of a user could be represented by the

following tuple: (*guarantor, "George Smith")*. There are no hardcoded limits on the number, length and characters that are usable in generic attributes.

Generic attributes (or tags [38]), from the point of view of applications, are attribute (name, value) pairs that can be assigned to VO users and that end up in the attribute certificate issued by voms (i.e., when a user issues a voms-proxy-init command). From the point of view of the application developer, a GA has a name, a description, a value and a context (or qualifier, described later on).

1. The *name* is the unique ID for the attribute.
2. The description may be used to provide information about the meaning and use of the attribute, and is bound to the attribute name.
3. The value is the actual value of the attribute for a specific user, and may be different for each user.

An example of a GA name and description could be:

attributeName	attributeDescription
emailAddress	This attribute contains the email address for a user

While the values of such an attribute may be different for each user:

userName	attributeName	attributeValue
andrea	`emailAddress`	andrea.ceccanti@cnaf.infn.it
valerio	`emailAddress`	valerio.venturi@cnaf.infn.it
vincenzo	`emailAddress`	**NULL**

In the current implementation, Voms-Admin (VA) provides the tools and interfaces to assign GAs to VO users. When a GA is assigned to a user, it ends up in the VOMS certificate generated for that user.

However it could be time-consuming and tedious for an administrator to assign attributes on a user-by-user basis. For this reason, VA and VOMS provide some "shortcuts", e.g. GAs may be assigned to VO groups and roles. When a GA is assigned to a group, it ends up in the VOMS credentials of all the users who are members of that group. Likewise, when a GA is assigned to a role within a group, it ends in the proxy of all the users who have that role in that group.

To avoid potential ambiguities when GAs with the same name, and even values, are defined in multiple contexts for a specific user (e.g. assigned by different administrators at different group hierarchy levels), VOMS provides a context qualifier to distinguish the different attributes.

For example, suppose an administrator assigns the GA (*A, 'aValueForAndrea'*) to user *andrea*, and the GA (*A ,'aValueForGroupTestG1'*) to group */test/g1*. This leads to user *andrea* having two GAs with the same name assigned to her, one assigned directly to the user herself and one assigned to her as a member of group */test/g1*. VOMS provides a context qualifier to distinguish these two attributes. More specifically, Andrea's proxy certificate will contain the triples,

```
('', A, 'aValueForAndrea'), ('/test/g1',A,'aValueForGroupTestG1')
```

where the first field is the qualifier, in which ' ' is the default used to mean "attribute assigned directly to user". The understanding of GA names and values is left to the applications that are willing to use this feature, i.e. VOMS does not infer any semantic meaning of a GA's content. In the context of this project, it means that if required, PERMIS policies could place some specific semantics or lack of semantics on GAs, for example, PERMIS could decide to always ignore the qualifier since it already supports and can distinguish between the values of multi-valued attributes.

- **FQAN**

A Fully Qualified Attribute Name (FQAN for short) is a compact way to represent a user's membership in a group, along with their role memberships, if any. Its general syntax is:

<groupname>/Role=<rolename>/Capability=NULL

where the /Capability=NULL may be omitted since capabilities are now a deprecated feature of VOMS.

For example, belonging to the group */test/italian* may be represented by the following FQAN: */test/italian/Role=NULL/Capability=NULL*, (or possibly by */test/italian/* in the future) while holding the role *sgm* in the same group will be represented by the following FQAN: */test/italian/Role=sgm/Capability=NULL or simply /test/italian/Role=sgm*

2.3 Related projects and middleware infrastructure

The VPMan project should be seen as an essential complement to work being carried out in several Grid middleware development projects. We consider the following projects to be of special interest: OMII-UK, gLite and Globus Toolkitv4. We provide a brief introduction of these other projects here.

2.3.1 OMII-UK

OMII-UK aims to provide software and support to enable a sustained future for the UK e-Science community and its international collaborators. OMII-UK aims to be the principal source for reliable, interoperable and open-source Grid middleware components, services and tools to support advanced Grid-enabled solutions in academia and industry.

For this reason, OMII-UK supports open-source software development by investing in community developers to produce the functionality required by the user community.

Drawing upon this software and other packages from the open-source developer community, OMII-UK provides a secure web service hosting environment, web services and the necessary tools and environments to access these services. The latest version of the OMII-UK software is OMII 3.4.0.

The OMII software focuses on the needs of distinct yet important stakeholders within Grid computing: the Service Provider and the Client.

The OMII server stack comprises mainly a secure web services container, in addition to other optional software component services. Users can interact with the OMII grid using either a command line tool or through any application using the supplied Java programming interface.

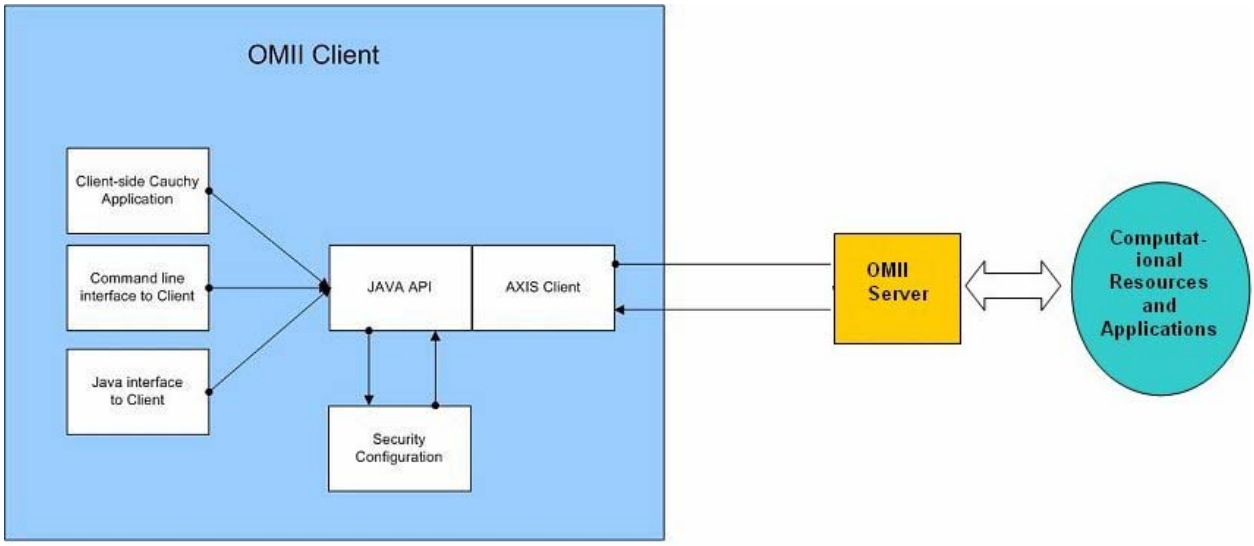

Figure 3. OMII-UK architecture

OMII-UK security is based on WS-Security. Developers can write standard web-services and benefit from the security and authorisation model of the OMII service provider.

An optional authorisation module - Process-Based Access Control (PBAC) can be used to enforce a finer grained security models. PBAC determines who can do what, and under what circumstances based on history of usage of a service.

OMII Authorization Service (OMII-AuthZ) [9] is a project at LeSC integrating PERMIS with the OMII-UK software environment. It aims at providing an access control mechanism for Web Services with per-operation/per-service granularity.

2.3.2 Globus Toolkit

The Globus Toolkit is an open source software toolkit used for building Grid systems and applications. It is being developed by the Globus Alliance and many others all over the world. A growing number of projects and companies are using the Globus Toolkit to unlock the potential of grids for their cause. Globus is arguably the de facto technology for Grids today with the latest release GT version 4 now based on web service technologies and standards.

PERMIS has been integrated into Globus Toolkit as an authorization decision entity within the project Grid API. VOMS has been integrated into Globus Toolkit as well (see section 4.2). This makes Globus a good starting point to integrate PERMIS and VOMS.

2.3.3 gLite

The gLite middleware comes from a number of Grid projects including: DataGrid [5]; DataTag [6]; Globus [7]; GriPhyN [8]; iVDGL [9]; EGEE and LCG. Currently, the glite middleware is maintained and developed in the context of the EGEE project that has a main goal of providing researchers with production level access to geographically distributed computing Grid resources.

The current release version is glite version 3 and comprises security services, information and monitoring services, data services, job management services, and helper services. These services

were developed to follow a service oriented approach, mostly based on web-services. They provide an open source implementation of the foundation services that are application independent and need to be deployed at all sites connected to the infrastructure.

On top of this foundation, an open-ended set of application specific higher-level services that can be deployed on-demand at specific sites are either provided directly by the project or can be integrated from other sources and projects

The figure below shows the relationship of Applications, Grid Services, and Grid Foundation Middleware. The Grid Foundation Middleware, in turn, is based on basic Grid tools like Condor and the Globus toolkit [42].

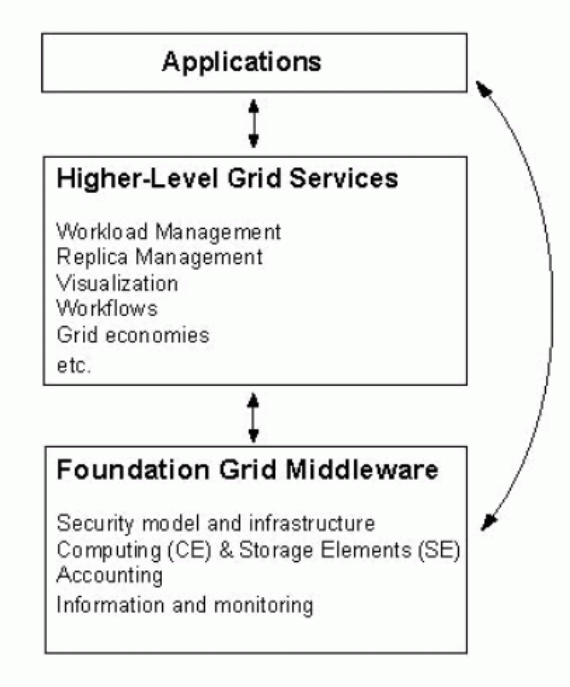

Figure 4. gLite levels

As indicated before, VOMS is part of the gLite security infrastructure. The authorisation of a user on a specific Grid resource can be done in two different ways. The first relies on the grid-mapfile mechanism. The Grid resource has a local grid-mapfile which maps users to local accounts. The second way relies on the Virtual Organisation Membership Service (VOMS) and the LCAS/LCMAPS mechanism, which allows for a more detailed definition of user privileges (more details below).

2.3.4 Shebangs

The Shebangs project is one of the projects which have demonstrated how the Shibboleth and Grid worlds can be harmonised. Shibboleth is standards-based, open source middleware software which provides Web Single SignOn (SSO) across or within organizational boundaries. The Shibboleth software implements the OASIS SAML v1.1 specification, providing a federated Single-SignOn and attribute exchange framework.

The SHEBANGS project provides Shibboleth Based Authentication for Grid Infrastructures by translating the credentials obtained by Shibboleth to Grid Security (GSI/VOMS) credentials understood by the National Grid Service (NGS) [10].

Users asking for VOMS credentials are redirected to an IdP be authenticated. The created VOMS credentials are stored in a MyProxy server to be used later by a portal to access resources.

2.3.5 GridShib

The GridShib project [46] is being carried out by the Globus team to integrate Shibboleth with the Grid, specifically proxy certificate based authentication with attribute based authorisation where the user's attributes are retrieved from their Shibboleth IdP. This will allow grid users who have used their proxy certificates to submit their jobs to the grid, to also pick up their attributes held by their local Shibbolized institution.

2.3.6 ShibGrid

The ShibGrid project is attacking Shibboleth integration with the Grid from the opposite end to GridShib. ShibGrid will allow NGS users, including those without UK e-Science X.509 public key certificates, to securely access the NGS resources, through the integration of Shibboleth, GSI and myProxy for user authentication. Users will then be able to access internal and external resources seamlessly using a single institutionally controlled identity [11].

2.3.7 OGF SAML AuthZ Grid API

The project integrated PERMIS with GT3 and GT4 [6] using the OGF SAML profile [17]. This SAML profile is the one being used by OMII UK as well. However, the SAML protocol has been shown to be deficient, and will need replacing by the GridAPI v2 based on XAMCL and WS-Trust

2.3.8 SIPS

This project integrated PERMIS with Shibboleth and Apache [7].

2.3.9 GridShibPERMIS

This project integrated PERMIS with GridShib [8] so that the attributes retrieved from the Shibboleth IdP can be used to authorise the user when the resource is controlled by a PERMIS policy.

2.3.10 NeSC Glasgow Shibboleth work

The National e-Science Centre at the University of Glasgow has shown through a variety of projects including DyVOSE, GLASS and ESPGrid how Grids can be accessed and used via Shibboleth where fine grained security is enforced through Shibboleth or portal configuration technologies. One recent project OMII SPAM-GP is developing a family of JSR-168 compliant portlets to scope attribute acceptance policies; attribute release policies; for dynamic portal content configuration and for pushing attributes to remote parties. This project has just started and will run for 1 year.

3. Complementary security system components

3.1 User side infrastructure components

3.1.1 VOMS UI

As indicated above, VOMS offers a User Interface that comprises a client (voms-proxy-init) and some ancillary utilities (voms-proxy-info, voms-proxy-destroy, voms-proxy-list). to generate a proxy

certificate that contains extra information about the user and the VOs they belong to.

Moreover, the user may consult (after being authenticated), through the web interface, their available roles on a VOMS server in the VOs they are involved in.

3.1.2 MyProxy

A significant enabling mechanism for GSI, that provides user mobility, is the availability of a service to manage user certificates. The MyProxy service [40] is open source software for managing X.509 Public Key Infrastructure (PKI) security credentials (certificates and private keys). In order to free the user from only operating from the workstation on which their private key is installed. MyProxy combines an online credential repository with an online certificate authority to allow users to securely obtain credentials when and where needed.

Storing credentials in a MyProxy repository allows users to easily obtain RFC 3820 proxy credentials, without worrying about managing the associated private key and certificate files. Users can use MyProxy to delegate credentials to services acting on their behalf (like a grid portal) by storing credentials in the MyProxy repository and sending the MyProxy passphrase to the service.

They can also use MyProxy to renew their credentials, so, for example, long-running jobs don't fail because of expired credentials. The MyProxy server can provide a more secure storage location for private keys than typical end-user systems. It can be configured to encrypt all private keys in the repository with user-chosen passphrases, with server-enforced policies for passphrase quality. By using a proxy credential delegation protocol, MyProxy allows users to obtain proxy credentials when needed without ever transferring private keys over the network.

However, MyProxy had not been completely adapted for VOMS support. Grix is a project that has now dealt with this issue [41]. Grix is a Java GUI application that enables, amongst other things, the creation of a MyProxy credential with or without a VOMS certificate.

3.1.3 Acacia

Acacia provides a Java implementation of various Attribute Certificate (AC) functionalities. In particular, it provides various client interfaces to the VOMS system, as well as an AC server that can act as a VOMS Server. It is currently maintained as a SourceForge project. Future developments for this project include supplying a command line interface to VOMS and the possibility of storing the produced proxy certificate (including VOMS credentials) in a MyProxy server.

At this time a Java WebStart interface is the only user interface supplied and it allows a user to upload a proxy certificate (including VOMS credentials) to a portal so that the VOMS attributes for that user can be extracted and used for authorisation purposes. The WebStart solution exists because currently proxy certificate support in browsers is minimal. This client allows a user to interact with the VOMS system as follows:

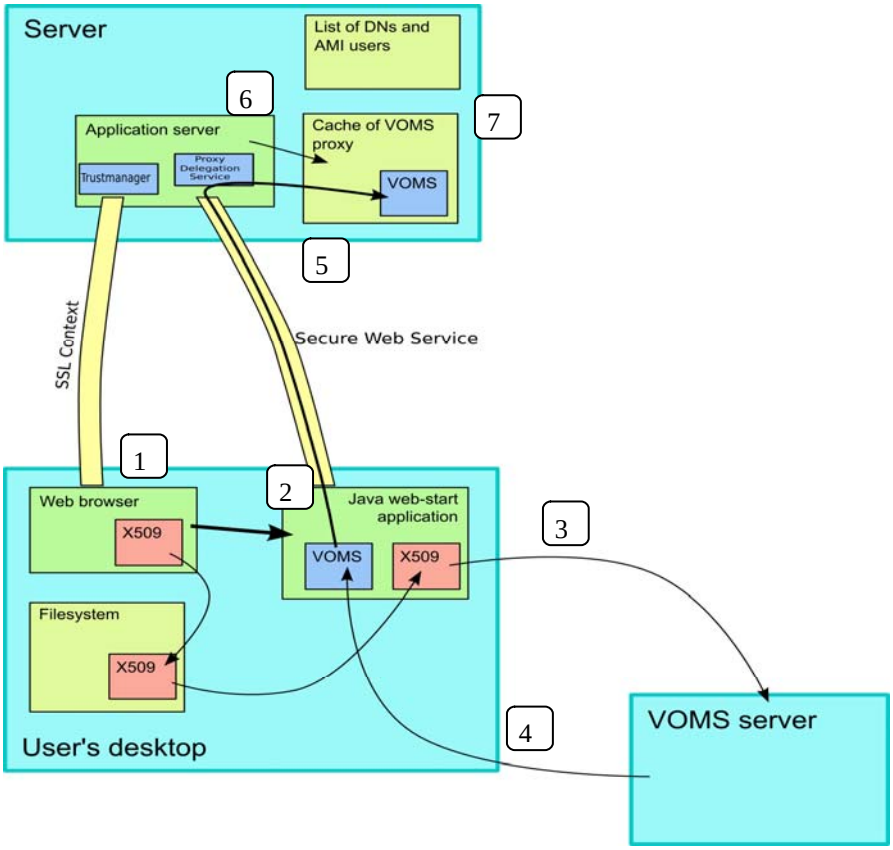

Figure 5. Acacia tool

The typical scenario of using Acacia is as follows:

(1) The grid users authenticate themselves to the portal using a grid certificate loaded into their browser.

(2) The portal allows the user to launch a Java web start application to create a proxy certificate (or to check its current validity) if they have been successfully authenticated.

(3) The user chooses their grid credential file to be used with the VOMS server along with a group, role and VO. This credential file (.p12) holds the both the users grid certificate and key. This file is loaded into a truststore so that it can be used to create a normal grid proxy certificate. This normal proxy is then used to identify the user to the VOMS server.

(4) The VOMS server sends back a VOMS Attribute Certificate in an appropriate X509 extension (VOMS proxy) to the Web Start application.

(5) Using Axis set up in Tomcat the VOMS proxy is uploaded to a server (to a given endpoint) via a secure web service attachment

(6) A proxy delegation service is available on the server.

(7) The extracted users VO, group and role can be used for authorization purposes within the portals local authorization mechanism in this case PERMIS.

There is an agreement between the Grix project and Acacia that MyProxy functionality can be integrated into the Acacia project so that Acacia will be able to push a generated user proxy to a variety of service end points, e.g. a Tomcat Axis web-service and MyProxy server.

3.1.4 Gridsite

The Gridsite solution doesn't require sending the proxy certificate via the browser; however, the web server has to fetch a "DN List" of authorised people, available from the VOMS server via HTTPS. Each of these DN lists define a group so if a persons' DN is within a certain list then they can be authorised as being part of that group. This GridSite system for example allows definition of access groups on www.gridpp.ac.uk. This fetching of lists has to be done usually between once per hour and once per day. A working VOMs solution that is based on the Gridsite framework depends on an Apache server and specifically the Apache module mod_ssl-gridsite.

3.2 Resource side infrastructure components

Many authorisation components are used in addition to PERMIS by grid infrastructures today. Here our focus is on GT4 authorization framework.

3.2.1 VOMS and GT4

GT4 introduces its own authorization framework [25]. The GT4 Java Web Services runtime invokes a series of message interceptors to process each message when it is first received (i.e., before it reaches the application). Two types of interceptors are of interest from an authorization perspective: Policy Information Points (PIPs) and Policy Decisions Points (PDPs).

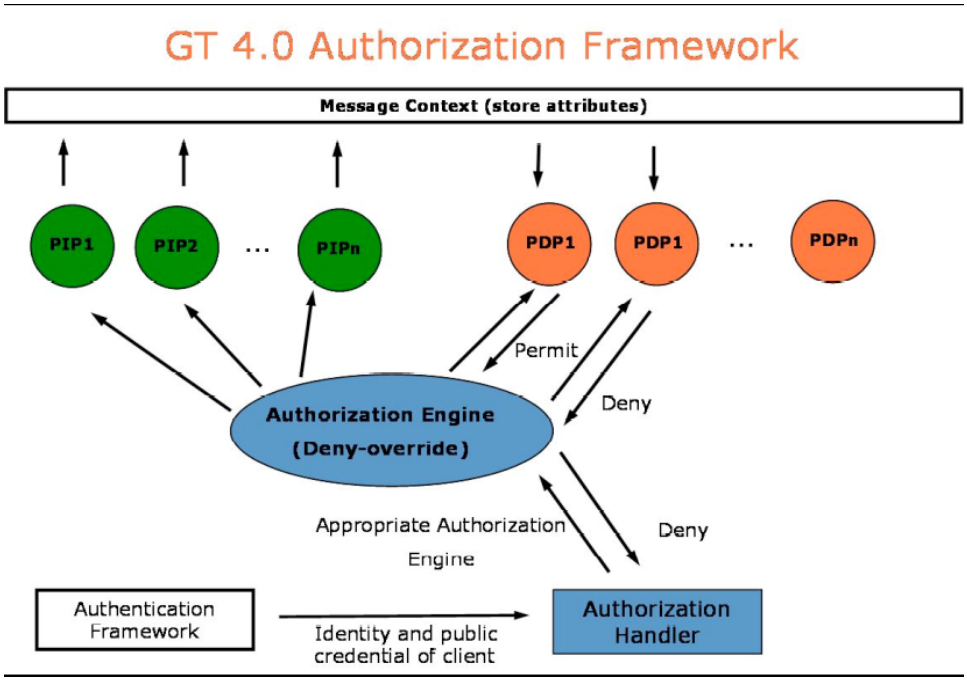

Figure 6. GT authorization framework

VOMS is integrated with the Globus Toolkit so that the user's credentials can be passed around with

the grid job, embedded inside the user's proxy certificate.

A GT4 VOMS PIP and PDP allow GT4 to access and process VOMS attribute certificates. The VOMS PIP parses VOMS attributes and stores them in the GT runtime. The VOMS PDP allows or denies requests based on these attributes and its configuration. The available VOMS PDPs are simple driven by configuration file. The authorization may be based on subject DN's (gridmap file) or accepted VOMs attributes, e.g.:

```
"/SOMEVO/SomeAttribute/Role=someRole/Capability=NULL"
"/SOMEVO/Role=NULL/Capability=NULL"
```

The PIP and PDP can be used together to allow or deny access to a service based on the requester's VOMS attributes.

The GT4 authorization framework enables other systems to be plugged in as well, e.g. the GridShib PIP which integrates SAML attributes. The GridShib authorization modules for GT support the simultaneous use of VOMS and SAML attributes (by optionally making a callout to the VOMS authorization modules).

3.2.1.1 Globus VOMS modules

When Globus VOMS modules [26] are installed by a user to a particular Globus installation, they are essentially just making Java libraries available to any service that needs them. In order to actually activate any authorization logic, the service configurations need to:

- list the VOMS modules in the authorization section of the relevant security descriptor
- include the VOMS module configurations, providing necessary per-module information (such as what VOMS servers to trust). The full list is below.

Like all authorization modules for the Java container, they can be used to protect the container as a whole, particular services, or particular WSRF resources.

3.2.1.2 Security descriptors

Security descriptors [27] contain various security properties like credentials, the gridmap file location, the required authentication and authorization mechanisms and so on. There are four types of security descriptors in the code base for setting container, service, resource and client security properties:

container security descriptor	determines the container level security requirements that need to be enforced.
service security descriptor	determines the service level security requirements that need to be enforced.
resource security descriptor	determines the resource level security requirements that need to be enforced.
client security descriptor	determines the security properties that need to be used for a particular invocation.

Figure 7. Security descriptors

Each of these is represented as an object and can be altered programmatically. Service and container security descriptors can be configured as XML files in the deployment descriptor as shown below. Resource security descriptors can only be created dynamically, either programmatically or from a descriptor file. A client security descriptor can be configured as an XML file and set as a property on the client Stub.

3.2.2 Grid map file

The *gridmap-file*, is a simple mechanism for allocating local user privileges to grid users. It is a simple list, resident at resource sites, which maps authorized grid users (expressed as Distinguished Names) to local credentials, e.g. usernames on Unix system.)

The format of the file is very simple: one line for each user which is allowed access. Each line has two fields separated by whitespace: the distinguished name and the local user account. For example, a gridmap file which maps a distinguished name to local account “bn29” could be the following:

Distinguished Name	Local account
"/O=Kent/OU=ISSRG/CN=Bassem"	bn29

3.2.3 PRIvilege Management and Authorization (PRIMA)

PRIMA [45] is an authorisation model and system. It has been implemented as an extension of the Globus Toolkit Grid middleware namely the Globus Gatekeeper with the Grid-map callout introduced in GT3.2 and Globus Web service gatekeeper (globus-ws) introduced in GT4.
In PRIMA, a privilege is a platform independent, self contained representation of a fine-grained right (subject, action, resource) signed by an issuer.
Resource administrators create and manage polices for their resources. They externalize privileges by delegating them to other users. The PRIMA representation uses an XACML construct to encode the externalized form of the privilege.

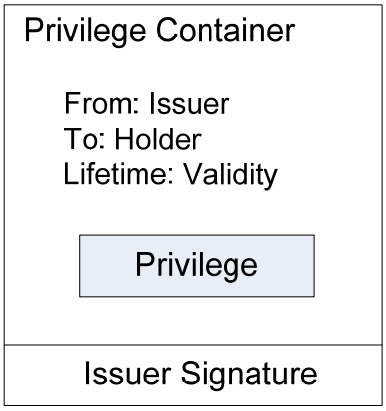

Figure 8. Privilege container

The holder of privileges can selectively provide individual privileges to grid resources when requesting access. The PRIMA PEP validates the supplied privileges and verifies that the issuer is authoritative to issue the specific privileges for the resources (calling the PDP for that). A set of valid privileges is then determined.
The PRIMA PDP is a policy engine for an XACML policy and is built using Sun’s java XACML library. It is queried to evaluate user requests against the set of valid privileges and the access control policies set up by the system administrators. The PDP returns an authorization decision along with a

set of instructions termed “obligations”, from the PDP to PEP, on how the requested service should be confined and monitored.

3.2.4 LCAS and LCMAPS

The Local Centre Authorization Service (LCAS) [33] is a site-local service that can authorise users based on their name, their VO affiliation, and the resources requested. In order to run jobs or store files within a traditional UNIX system, the Local Credential Mapping Service (LCMAPS) can make sure that user requests are sandboxed in local accounts with unique group memberships. Such accounts can span a machine or a cluster, or an entire administrative domain.

To keep track of tasks sent to the fabric, the relation between the identity and authorization tokens presented on the Grid side, and their mapping into local credentials (unix groups, account names, etc), the Job Repository (JR) was developed. Based on a backend ODBC-interface, it is a database containing this essential information.

LCAS, LCMAPS and the JR were developed in the context of the EU DataGrid project, and are now also incorporated into gLite (described previously).

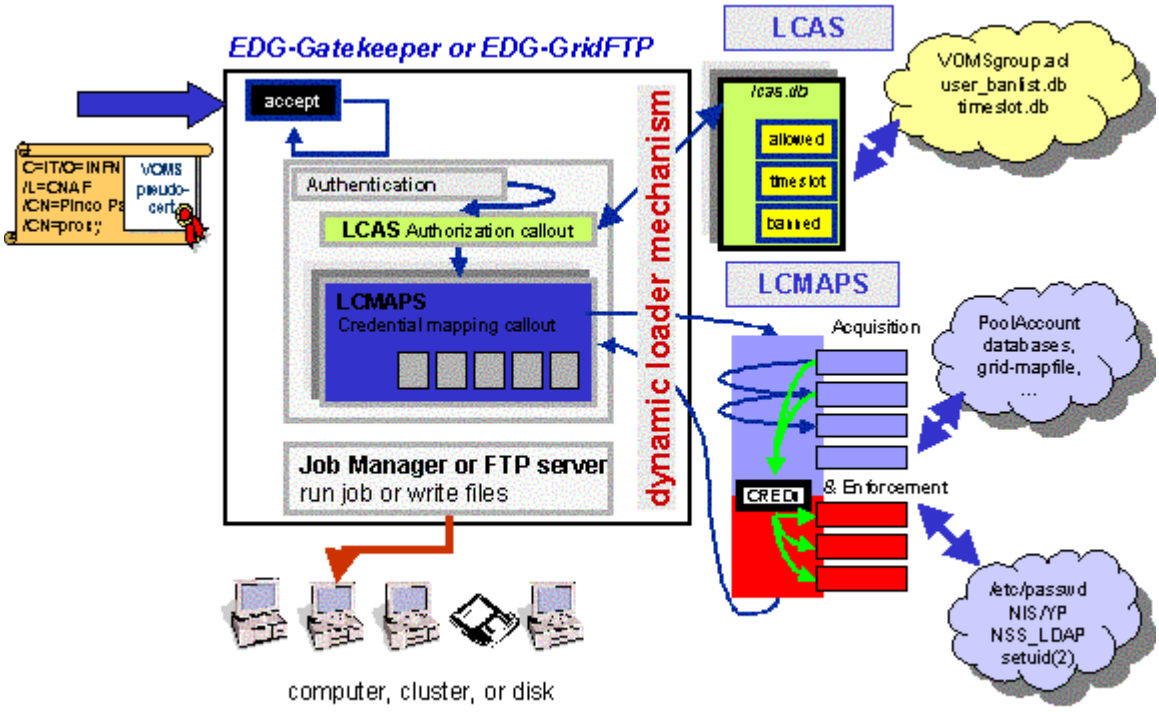

Figure 9. LCAS+LCMAPS

3.2.4.1 LCAS

LCAS [33] is the authorisation decision engine that adds access controls to the "gatekeeper". The gatekeeper accepts job requests from external sources (like the end user or the workload management system) over an authenticated channel. The concept is that different independent authorisation modules may be plugged-in to LCAS, thus creating a flexible system. The plug-in framework enables multiple independent authorisation modules to collectively grant or deny access to the resource. The decision is based on the requested resources (expressed via the Resource Specification

Language RSL), the identity of the requester, and the authorisation credentials presented by the end-user in the proxy certificate. If VOMS is used this will be the VO, Group and Role combinations the user has acquired from VOMS. A basic policy language allows selection of which authorisation plug-ins are to be invoked, and the access decision is the logical "and" of the answers of the individual plug-ins. So it would be fair to say that LCAS performs the same functionality for GT2 that the GT4 authorisation framework provides, in that it is configured with the various PDPs to be called in which order, and it then combines the results of their decisions.

LCAS is a shared library integrated into the Globus Toolkit v2 (pre-webservices), which is written in C, and called when an authorization decision is needed. LCAS provides three default authorization modules, also written in C: lcas_userallow.mod, lcas_userban.mod, and lcas_timeslots.mod. An additional optional plugin lcas_voms.mod decides if the user is authorized based on the VOMS credentials stored in the user's proxy X509 certificate. It therefore would be possible to add another module to LCAS, say lcas_permis.mod, that calls the PERMIS decision engine from a C program (in a similar way to which Shibboleth/Apache calls PERMIS via mod_permis).

- lcas_userallow.mod module checks if the user is allowed on the fabric (currently the gridmap file is checked). This plugin checks a file that contains a list of DNs (subjects of the X509 certificates) of allowed users. If the DN of the user for which the authorization request is made is found in the list, the plugin grants access to the site.

- lcas_userban.mod module checks if the user should be banned from the fabric. This plugin checks a file that contains a list of DNs (subjects of the X509 certificates) of users to be banned from the site. If the DN of the user for which the authorization request is made is found in the list, the plugin denies access to the site.

- lcas_timeslots.mod module checks if the fabric is open at this time of the day for datagrid jobs. This plugin makes an authorization decisions based on available time slots. Currently it reads a text file that contains the available time slots of the form

```
# If the fabric is open on working days from 8:30-18:00 h, from 1 July 2002 till 15 January 2003
 30-0        8-18       1-15       7-1       2002-2003      1-5
```

All three modules get their information from simple configuration files: allowed_users.db, ban_users.db and timeslots.db respectively.

In addition a plugin is provided that decides if the user is authorized based on the VOMS (VO Membership Service) information stored in the user's proxy X509 certificate:

- lcas_voms.mod plugin forms the link between the VOMS data found in the user's grid credential (X509 certificate) and the LCAS system. It retrieves the VOMS data through the VOMS API. In order to control user access to the site, the VOMS data will be checked against either:

  1. a text file containing just a list of allowed VO-GROUP-ROLE combinations

```
"/VO=wilma/GROUP=wilma/*"
"/VO=fred/GROUP=fred/*"
```

  2. a policy file in GACL [36] format (GACL is an XML Access Control Language).

```
<?xml version="1.0"?>
<gacl version="0.0.1">
<entry>
<voms-cred>
<voms>/O=dutchgrid/O=hosts/OU=nikhef.nl/CN=asen.nikhef.nl</voms>
<vo>wilma</vo>
<group>wilma</group>
</voms-cred>
<allow><read/><write/></allow>
<deny><list/></deny>
</entry>
<entry>
<person>
<dn>/O=dutchgrid/O=users/O=knmi/CN=Wim Jan Som de Cerff</dn>
</person>
<allow><read/></allow>
<deny><list/></deny>
</entry>
</gacl>
```

3. a policy file in XACML [21] format (the OASIS access control policy language)

External parties can develop their own plug-ins to provide additional functionality, without the need to re-compile the LCAS framework software. LCAS reads from a configuration (lcas.db) the plugins that it should load along with their configuration parameters. For example:

```
pluginname="lcas_userallow.mod",pluginargs="allowed_users.db"
pluginname="lcas_userban.mod",pluginargs="ban_users.db"
pluginname="lcas_timeslots.mod",pluginargs="timeslots.db"
pluginname="lcas_plugin_example.mod",pluginargs="Some bogus arguments"
pluginname="lcas_voms.mod",pluginargs="-vomsdir /etc/grid-security -certdir /etc/grid-
security/certificates -authfile /opt/edg/etc/lcas/lcas_voms.gacl -gacl_use_voms_dn always"
```

We envisage plugging a PERMIS module into LCAS to decide if a request should be allowed or not according to the PERMIS policy and the VOMS attributes extracted from the proxy certificate. PERMIS authorisation is more advanced than most of the methods shown here (apart from the XACML PDP) and enables enforcement of many more features (see section 2.1).

3.2.4.2 LCMAPS

LCMAPS [34] is the local credential mapping service. It is a pluggable system that controls how an incoming grid request obtains a User Id (UID) and Group Id (GID) on the system.

LCMAPS can load and run one or more 'credential mapping' plugins. The use of a plugin-framework architecture for LCMAPS makes it very easy for sites/organizations to add new functionality to LCMAPS by writing new plugins. The LCMAPS framework consists of the following components:

1. The ***plugin manager***, which is responsible for managing, loading and running the LCMAPS plugins.

2. The ***evaluation manager***, which is responsible for the order in which the LCMAPS plug-ins are called. In fact, it is responsible for creating an appropriate environment based on the users set of credentials. In order to accomplish this, the system must know about policies for

assigning resources to a user or service. The proposed policy language is based on graphical representation, translated into text format [35]. For example, according to which VO a user is a member of, the system is able to tell how much disk space may be used, how many processors are reserved for the VO, billing, etc (even if it is mapped to a local account). The evaluation manager is driven by this policy.

Based on the user's global credentials (more specifically the user's X509 certificate) and the job specification (JDL), the LCMAPS plugins have to perform either of these two tasks:

- Acquire local credentials (e.g. uids, gids).
- Enforce (apply) the local credentials.

The following LCMAPS plugins are available for acquiring credentials:

1. non-VOMS-aware plugins, e.g.:
   - lcmaps_localaccount.mod: this plugin maps the user's DN into a the local account name using a gridmap file.
   - lcmaps_poolaccount.mod: this plugin maps the user's DN into a pool account name using a gridmap file.
2. VOMS-aware plugins that use VOMS attribute in the user certificate for the credential mapping.
   - lcmaps_voms.mod: this plugin extracts the VOMS information from the user's X509 proxy certificate.
   - *lcmaps_voms_localaccount.mod*: this module tries to find a local account (UID) based on the VO information in a gridmapfile.
   - lcmaps_voms_localgroup.mod: this plugin tries to find a local group Ids (GIDs) based on the VO information and a groupmapfile which is similar to the gridmapfile, only the groupmapfile defines the *group Id* (i.e. GID) for a particular set of users in a VO.
   - lcmaps_voms_poolgroup.mod: this plug-in tries to find a pool group Ids (GIDs) based on the VO information and a groupmap file.
   - lcmaps_voms_poolaccount.mod: this plug-in tries to find a pool account based on the VO information and a gridmapfile.

The following plug-ins are available for enforcing credentials:

- lcmaps_posix_enf.mod sets the real and effective user and group ID for the current process. The (BSD/POSIX) functions setreuid(), setregid() and setgroups() are used to change the privileges of the process from root to that of a local user.
- lcmaps_ldap_enf.mod updates a fabric-central user directory for userid and groupid information. It alters the user and group settings in the ldap database [43], using the user and groups settings provided by the credential acquisition plugins.
- lcmaps_afs.mod is an Acquisition and Enforcement Plugin that maps the DN onto local Kerberos and AFS tokens [44].

LCMAPS reads in the file lcmaps.db, the options for the various plugins and the policies, i.e. the order that the plugins should be executed.

As described, LCMAPS is dedicated to acquiring and enforcing local credentials for accomplishing the required tasks. PERMIS may supply an equivalent functionality of acquiring the local credentials by returning them via obligations. We may have a PERMIS policy that indicates if a particular VO member is authorised to run a particular task, and if so, to indicate, for instance, a UID and GID to be used. LCMAPS has already specialised plug-ins for enforcing these credentials, however, PERMIS doesn't indicate how obligations are enforced by the system, although a separate Obligations Service

will be provided that will return the obligated local user Ids.

3.3 VOMS administration

We present here the administrative tools available for managing the system (VOMS, users, roles, accounts). Note that in VPMan we are less concerned with how VOMS is managed, since we focus on the authorisation process of users who have already acquired VOMS credentials.

3.3.1 VOMS-ADMIN

Voms-admin constitutes the administrative tool for VOMS whose basic functionality is implemented in C++ by INFN [29]. The extended functionality is implemented as a web service, with command line and web interfaces:

- **Admin:** provides the administrative functionality.
- **Compatibility:** provides access to the user list for *gridmap-file* generation.
- **Request:** handles user requests for administrative events and provide a simple framework for administrators to process them.
- **History:** provides a lookup functionality of current and past events, to answer questions like *"was this user a member of my VO last summer?"*
- **Core:** provides the basic functionality for users.

3.3.2 VOMRS

VOMRS [30, 31] stands for Virtual Organization Management Registration Service and is another service used with VOMS. Its main purpose is to offer a comprehensive set of services facilitating secure and authenticated management of VO membership, grid resource authorization and privileges.

VOMRS implements multiple features on the top of the basic *voms-admin* API. For example, it implements a registration workflow providing means for collaborators to register in a Virtual Organization (VO), provides email notifications of selected events, supports VO-level control over which set of Certificate Authorities (CA) are trusted and permits delegation of responsibilities within the various VO administrators. It is also capable of interfacing to third-party systems and pulling or pushing relevant member information from/to them.

Through VOMS admin APIs, any modification and reorganization of the VO membership are propagated to VOMS. A clear picture summarizing the system is given in figure 10.

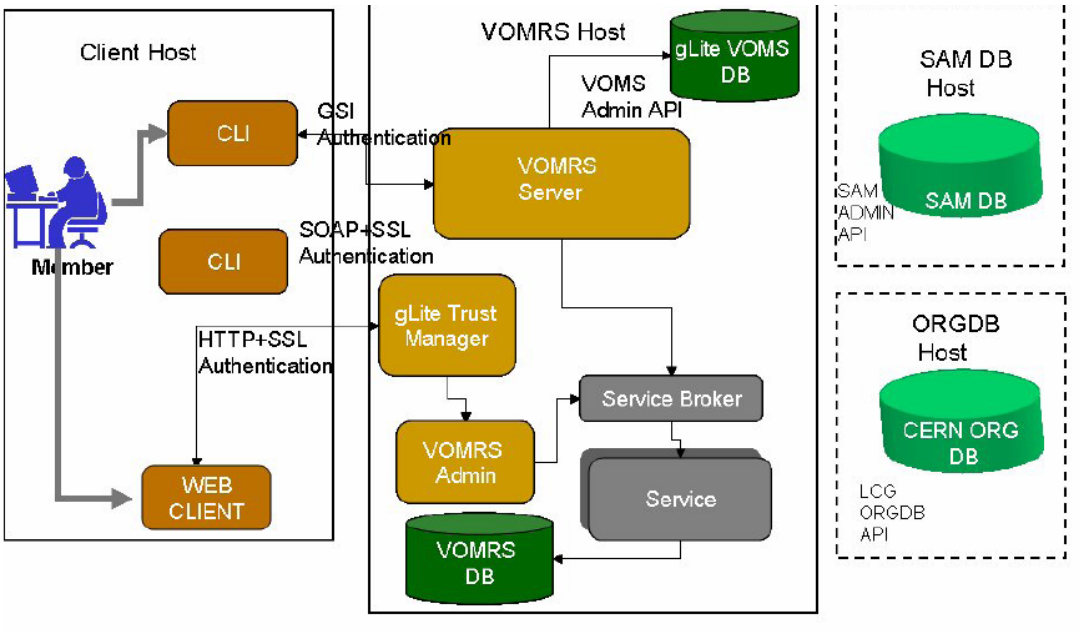

Figure 10. VOMRS management system

3.3.3 Grid User Management Systems (GUMS)

The GUMS system [32] proposes an alternative architecture to VOMS for user authorisation within a virtual organisation. The user registers once for a virtual organisation and this information is stored in a VO database server (which could be a VOMS server, but it does not have to be). A site daemon (Cron Job in Fig 11) pulls user information from this database and invokes local tools that track and manage the status of the user with respect to a local user account. According to local policies (user info, banned user, name mapping etc) the user may be accepted and thus a local account is created for him and the local gridmap file is updated with his DN (see Figure 11).

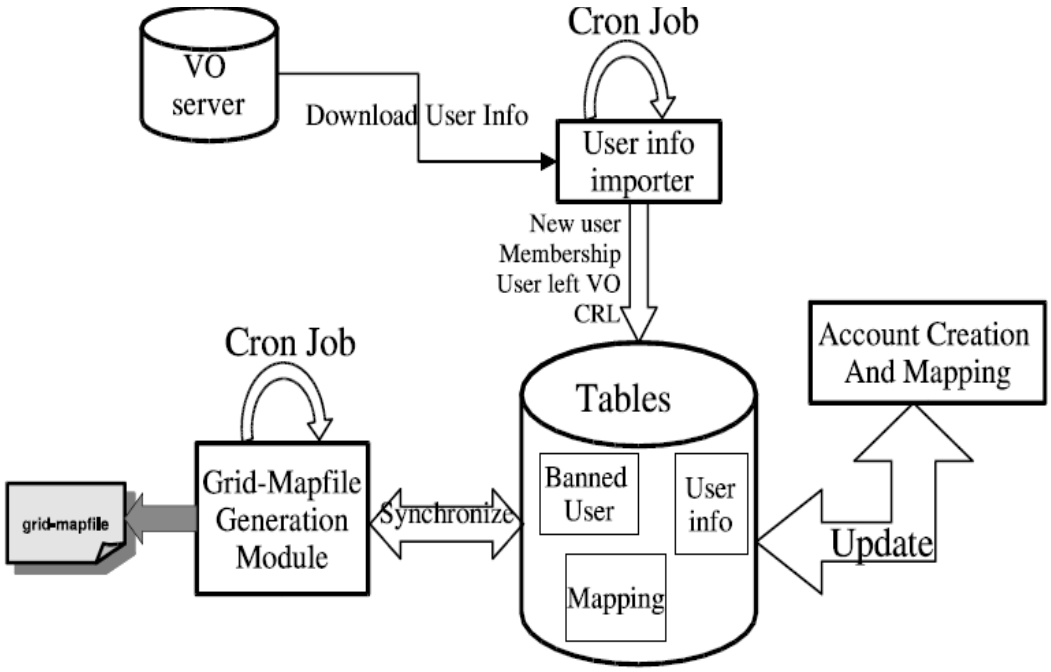

Figure 11. Grid User Management System architecture

GUMS may work with VOMS, in which case GUMS may poll VOMS Admin periodically to update its local list of users. In another way of functioning, GUMS may do dynamic role/group-based mappings according to the credentials pushed by the user, which may include his VOMS attributes.

4. Requirements

Though PERMIS and VOMS were included in the security infrastructures of many middleware, there is still work to do in order to seamlessly integrate VOMS and PERMIS within the same infrastructure. This is the motivation behind the VPMan project and constitutes the main part of its requirements. These requirements are:

1- Allow resources already protected by PERMIS to accept and make decisions based on VOMS ACs. PERMIS should be able to make authorization decisions based on the VOMS attributes: VO, group, subgroup, role and generic attributes (Gas).

2- PERMIS should be able to function with VOMS in either pull and push mode, though the former will be dependent upon enhancements to the VOMS server which are being carried out by INFN and are out of scope of this project.

3- The integration work in VPMan must not require any changes to VOMS or VOMS ACs which would prevent interworking with current VOMS's systems (although extensions for additional functionality may be OK).

4- The integration work with OMII-UK and the NGS shouldn’t require major changes in their security infrastructures.

5- The integration should take into consideration compatibility with different grid middleware especially pre-GT4 releases.

6- VPMan should allow resources owners who currently use VOMS authorization to implement/enhance their local policies using PERMIS tools. This includes offering a user-friendly policy editor adapted to the VO and particularly to VOMS terminology (VO, group, subgroup, role).

7- VPMan should also permit the integration of the security components in order to support the defined use cases in the accompanying document D1.2.

5. Conclusion

Both VOMS [1] and PERMIS [2] provide security management infrastructures for Grids but have addressed the problem from different ends of the spectrum. VOMS has concentrated on the management of user credentials, whilst PERMIS has concentrated on policy based decision making. Each has its strengths and weaknesses and their combination will be a powerful solution to Grid security management.

We believe that integrating VOMS and PERMIS, more specifically the VOMS user management and attribute assignment function with the PERMIS policy based authorisation decision function is of great benefit to the grid community including end users, service providers and administrators. Moreover, the integration work is motivated by use-cases being provided by the project partners.

6. References

[1] Alfieri, R., Cecchini, R., Ciaschini, V., Dell'Agnello, L., Frohner, A., Lorentey, K., Spataro, F., *From gridmap-file to VOMS: managing authorization in a Grid environment*, Future Generation Computer Systems. Vol. 21, no. 4, pp. 549-558. Apr. 2005

[2] D.W. Chadwick, A. Otenko *The PERMIS X.509 Role Based Privilege Management Infrastructure*. Future Generation Computer Systems, 936 (2002) 1–13, December 2002. Elsevier Science BV.

[3] The Globus Alliance, http://www.globus.org/toolkit/

[4] The Open Middleware Infrastructure Institute UK, http://www.omii.ac.uk/

[5] The Shibboleth project, http://shibboleth.internet2.edu/

[6] R.O. Sinnott, A. Asenov, D. Berry, S. Furber, C. Millar, A. Murray, S. Pickles, S. Roy, A. Tyrell, M. Zwolinski, *Meeting the Design Challenges of nanoCMOS Electronics: An Introduction to an EPSRC Pilot Project*, UK e-Science All Hands Meeting, Nottingham UK, September 2006.

[7] W. Xu, D.W. Chadwick, A. Otenko. *Development of a Flexible PERMIS Authorisation Module for Shibboleth and Apache Server*, Proceedings of 2nd EuroPKI Workshop, University of Kent, July 2005

[8] Chadwick, D.W., Novikov, A., Otenko, O. *GridShib and PERMIS Integration*. Terena Networking Conference (TNC 2006), Scicily, May 2006.

[9] See http://www.omii.ac.uk/downloads/project.jsp?projectid=68

[10] The Shebangs project, http://www.mc.manchester.ac.uk/research/shebangs

[11] ShibGrid project, http://www.oerc.ox.ac.uk/activities/projects/index.xml?ID=ShibGrid

[12] ISO 9594-8/ITU-T Rec. X.509 (2001) The Directory: Public-key and attribute certificate frameworks

[13] The VOMS Attribute Certificate Format
http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.ogsa-authz/docman.root.attributes/doc13797

[14] R.O. Sinnott, D.W. Chadwick, *Experiences of Using the GGF SAML AuthZ Interface*, Proceedings of UK e-Science All Hands Meeting, September 2004, Nottingham, England.

[15] D.W. Chadwick, L. Su, R. Laborde. *Use of XACML Request Context to access a PDP*. OGSA-Authz WG Draft Standard. 28 March 2006.

[16] D. Chadwick, G. Zhao, A. Otenko, R. Laborde, L. Su, T.A. Nguyen. *Building a Modular Authorization Infrastructure*, presented at All Hands Meeting, Nottingham, Sept 2006.

[17] Von Welch, Rachana Ananthakrishnan, Frank Siebenlist, David Chadwick, Sam Meder, Laura Pearlman. "Use of SAML for OGSI Authorization", GFD.66. March 2006, Available from http://www.ggf.org/documents/GFD.66.pdf

[18] D. W Chadwick, A. Otenko and T.A. Nguyen. *Adding Support to XACML for Dynamic Delegation of Authority in Multiple Domains*, to be presented at IFIP CMS 2006, October 2006

[19] S. Brostoff, M. A. Sasse, D. Chadwick, J. Cunningham, U. Mbanaso, A. Otenko. *R-What? Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists* Software: Practice and Experience, Volume 35, Issue 9, Date: 25 July 2005, Pages: 835-856

[20] OASIS. *Assertions and Protocol for the OASIS Security Assertion Markup Language* (SAML) V2.0, OASIS Standard, 15 March 2005.

[21] OASIS. *eXtensible Access Control Markup Language* (XACML) Version 2.0, OASIS Standard, 1 Feb 2005

[22] See http://middleware.internet2.edu/signet and http://middleware.internet2.edu/dir/groups/grouper

[23] Chadwick, D.W., *Delegation Issuing Service*, in NIST 4th Annual PKI Workshop, pages 62-73, Gaithersberg, USA, April 2005.

[24] See: http://littleblue.cnaf.infn.it/twiki/bin/view/VOMS/WebArchitecture

[25] See: http://www.globus.org/alliance/events/sc06/AuthZ.pdf

[26] Globus VOMS plugin http://dev.globus.org/wiki/VOMS

[27] See: http://www.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html

[28] VOMS guide, https://edms.cern.ch/file/571991/1/voms-guide.pdf

[29] VOMS administrative interface, http://edg-wp2.web.cern.ch/edg-wp2/security/voms/

[30] VOMRS guide, http://www.uscms.org/SoftwareComputing/Grid/VO/vox.pdf

[31] VOMRS functionality, https://twiki.cern.ch/twiki/bin/view/LCG/VomrsFunctionality

[32] See: http://www.atlasgrid.bnl.gov/testbed/gums/,
http://www.atlasgrid.bnl.gov/testbed/gums/VO_AAA_new.pdf and
http://grid.racf.bnl.gov/GUMS/guide_architecture.html

[33] See:
http://glite.web.cern.ch/glite/packages/R3.0/R20060502/doc/installation_guide_3.0-2.html
http://grid-it.cnaf.infn.it/fileadmin/sysadm/voms-integration/voms-integration.html#SECTION00043000000000000000
http://wiki.ngs.ac.uk/index.php?title=LCAS

[34] See:
https://twiki.cern.ch/twiki/bin/viewfile/LCG/LhcbPage?rev=1;filename=A_quick_introduction_to_VOMS.pdf
http://grid-it.cnaf.infn.it/fileadmin/sysadm/voms-integration/voms-integration.html#SECTION00043000000000000000
[35] PDL module, see: http://www.dutchgrid.nl/DataGrid/wp4/lcmaps/edg-lcmaps-0.0.3/pdl_requirements.pdf

[36] GACL, see: http://www.gridpp.ac.uk/authz/gacl/

[37] D.W. Chadwick, S. Anthony. "Using WebDAV for Improved Certificate Revocation and Publication" Paper submitted to Europki 2007 conference

[38] Vincenzo Ciaschini, Valerio Venturi, Andrea Ceccanti. "The VOMS Attribute Certificate Format". OGF OGSA Authz WG draft, Sept 2006. Available from
http://forge.gridforum.org/sf/go/doc13797?nav=1

[39] See: http://ref.web.cern.ch/ref/CERN/CNL/2003/001/DataTAG/

[40] MyProxy Credential Management service, See: http://grid.ncsa.uiuc.edu/myproxy/

[41] Grix project, See: http://www.grid.apac.edu.au/repository/trac/grix/

[42] See: https://twiki.cern.ch/twiki/bin/view/EGEE/EGEE2JRA1Mandate

[43] RFC 2307, "An Approach for Using LDAP as a Network Information Service", See
http://www.faqs.org/rfcs/rfc2307.html

[44] AFS system, http://www.psc.edu/general/filesys/afs/afs.html

[45] Markus Lorch, Dennis Kafura, "The PRIMA Grid Authorization System", Journal of Grid Computing, March 24, 2005

[46] Tom Barton, Jim Basney, Tim Freeman, Tom Scavo, Frank Siebenlist, Von Welch,
Rachana Ananthakrishnan, Bill Baker, Kate Keahey. "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy". Presented at NIST PKI Workshop, April 2006.

