Authorization and Account Management in the Open Science Grid

Markus Lorch, Dennis Kafura, Ian Fisk, Kate Keahey, Gabriele Carcassi,
Tim Freeman, Timur Peremutov, Abhishek Singh Rana

***Abstract*—An attribute-based authorization infrastructure developed for the Open Science Grid is presented. The infrastructure integrates existing identity-mapping and group-membership service using concepts prototyped in the PRIMA system. Authorization scenarios for requests to compute and data resources are detailed. A new SAML obligated authorization decision statement is introduced that attaches an XACML obligation to the authorization decision. The use of obligations enables site-centralized, service-independent policy management. Authorization decisions are enforced via a Workspace Service that creates constrained execution environments configured in accordance with the obligations and other attribute-based information. Finally, an experimental PRIMA authorization service that extends and simplifies the infrastructure is described.**

***Index Terms*—Authorization, Account Management, Attributes, Roles**

I. Introduction

A computational Grid supporting large-scale collaborative scientific research is organized around the concept of a "virtual organization." A virtual organization (VO) represents the resources and people that are intended to be part of the collective enterprise. Agreements among the parties participating in the VO define the rules for resource usage, the privileges that individuals within the VO are allowed to exercise, and how the VO itself is organized and managed. The security mechanisms enforcing the proper use of the resources contributed to the VO must confront the varying ways in which the VO members might be identified by their "real" home organization and the heterogeneity of resource types within the VO.

A VO may support a complex set of relationships defining which users are part of which projects within the VO and which users are designated to perform distinguished roles within the VO at various times. Examples of these relationships include:

- A single user may be a member of several projects. Not only are there different resource allocations for these projects, but the resource usage must be properly charged to the correct project and the correct VO.
- A single user may have multiple roles in a VO. At times the user may act as a project administrator and at other times the user acts a regular VO member. The "community service" resource usage when acting as the project administrator should be accounted for separately from the usage when acting as a regular VO member. Furthermore, when acting as the administrator, the user may have privileges not available to that same user when acting as a regular VO member. For example, when acting as the administrator the user may be able to terminate jobs running within the VO that were created by other users. This privilege is not available when the user is acting as a regular project member.
- A group of individuals may alternate the administration of the VO with only one individual at a time acting as the administrator. To insure non-interfering administration of the VO, the administrator function may be permitted to a given individual only during a pre-determined period of time.

These relationships are based on the personal experiences of one or more of the authors or are goals of the Open Science Grid project.

Current Grid security mechanisms exhibit several weaknesses when attempting to cope with the complex VO structure illustrated above. First, enforcement mechanism at the resource level may not be aware of VO groups or roles.

Manuscript received June 3, 2005. This work was supported in part by Fermi National Accelerator Laboratory, the Virginia Commonwealth Information Security Center, and the IBM Corporation.

M. Lorch. Author was with the Department of Computer Science at Virginia Tech, Blacksburg, Virginia 24060 USA, and has recently joined IBM Germany (e-mail: lorchm@acm.org).

D. Kafura. Author is with the Department of Computer Science at Virginia Tech, Blacksburg, Virginia 24060.USA (phone: 540 231 5568, fax: 540 231 6075, e-mail: kafura@cs.vt.edu).

I. Fisk. Author is with Fermi National Accelerator Laboratory, Batavia, IL, 60510, USA, (e-mail: ifisk@fnal.gov)

K. Keahey. Author is with Argonne National Laboratory, Chicago, IL 60439 USA, (e-mail: keahey@mcs.anl.gov).

G. Carcassi. Author is with Brookhaven National Laboratory, Upton, NY 11973, USA, (e-mail: carcassi@bnl.gov).

T. Freeman. Author is with University of Chicago, Chicago, IL 60637 USA, (e-mail: tfreeman@cs.uchicago.edu).

T. Perelmutov. Author is with Fermi National Accelerator Laboratory, Batavia, IL, 60510, USA, (e-mail: timur@fnal.gov)

A. S. Rana. Author is with the University of California, La Jolla, CA, 92093, USA, (e-mail: rana@fnal.gov)

Typically VO group and role policies are not communicated to the Grid resources. Resources therefore have no basis for differentiating between users from a given VO. As more users, applications, resources, and services are added to the VO, the level of security becomes increasingly more hazard-prone with respect to resource security, data and job security, accountability, and efficiency. A more flexible authorization mechanism is required that can distinguish between individual users and between the roles an individual user can hold. To achieve and maintain good service with the enhanced security, it is important to incorporate the policies of both the VO and the individual resources when defining the details of such an authorization system.

Second, multiple users are frequently mapped to the same resource-level account (e.g., all users from a given VO are mapped to a single, shared user account). This many-to-one mapping reduces the administrative overhead of manually maintaining individual user accounts and simplifies the sharing of (data) resources among members of a single VO. However, this mapping creates two major security problems. First, every access is granted with the full set of access privileges that the VO as a whole is authorized to assume. Second, the user activities are not well insulated from each other; users may inadvertently modify or destroy the work of other users mapped to the same account and tracing a problem to a particular client may not be possible. Overall, the many-to-one mapping provides limited support for the implementation and management of authorization policies and affords a relatively low degree of system security.

Third, site-level policies are typically replicated such that each Grid resource has its local copy of these policies (e.g., local grid-map file). In addition individual services often require policies to be stored in a proprietary format (e.g., the dCache kpwd file vs. the Globus grid-map file) while fundamentally containing the same information. The maintenance and auditing of these redundant policy sources is an administrative nightmare and often presents a weak component of the deployed Grid authorization system. A site-centralized, Grid service independent management system for such policy information can improve the maintenance and promote the consistent enforcement of site access control policies.

To address these weaknesses the following requirements have been defined for the Grid security infrastructure of the U.S. CMS and U.S. ATLAS high-energy physics collaborations:

- Resource providers (sites) define authorization policy based on groups and roles of the supported VOs, and have mechanisms that can enforce these policies consistently over all the resources in their domain.
- Enforcement mechanisms must support existing applications and use cases.
- The access rights with which a specific access is granted are reduced and ideally represent a fair approximation of the least amount of privileges required for this access.
- Users can drive/customize the allocation of a subset of their access rights to a specific access (e.g. through the selection of their current "role").
- Users may be a member of multiple collaborations (VOs) and different sub-groups within a collaboration. The system should support the separation-of-duties principle for these users.
- Users are to be separated from each other and an individual user's files must be protected against accidental or malicious modification by other users (including other members of the same VO).
- The management of local user account mapping tables (resource policies) should be improved and unified among compute and storage services.

The Privilege Project, a collaboration among Fermi National Accelerator Lab, Virginia Tech, and Brookhaven National Lab with developers from US-CMS, US-ATLAS and the Particle Physics Data Grid (PPDG), has developed an attribute-based authorization infrastructure to address these requirements based on previous work on PRIMA [1].

The developed software components are included in the latest releases of the Virtual Data Toolkit, a Grid middleware distribution, and are part of the Open Science Grid middleware.

The work described in this paper integrates the use of authorization attributes, flexible account management, and obligations to meet the defined requirements. Authorization attributes empower VOs to drive varying policies for different user groups and roles in order to limit which tasks the users can perform and with what priorities, on an access-by-access basis. The user is also empowered to select the appropriate VO/group/role combination according to the activity they plan to perform. Flexible account management is used to separate the activities of users from each other and to separate the activities of a single user acting in different roles. Finally, obligations are used as a critical element in constructing a site-centralized and unified service that avoids redundant storage of security information. The combination of these elements leads to a security structure within which computing and storage resources are empowered to intelligently enforce priorities and data access rights set at the VO level. Users are individually recognized and their activities controlled as needed in order to adhere to the site-specific security requirements.

The remainder of this paper is organized as follows. Section II discusses the authorization model and general architecture for compute resources in detail. Section III focuses on the slightly more complex authorization architecture for storage resources and Section IV discusses the use of obligations in authorization decision. Execution environments are discussed in Section V and an alternative policy decision point is covered in Section VI. A summary concludes the paper.

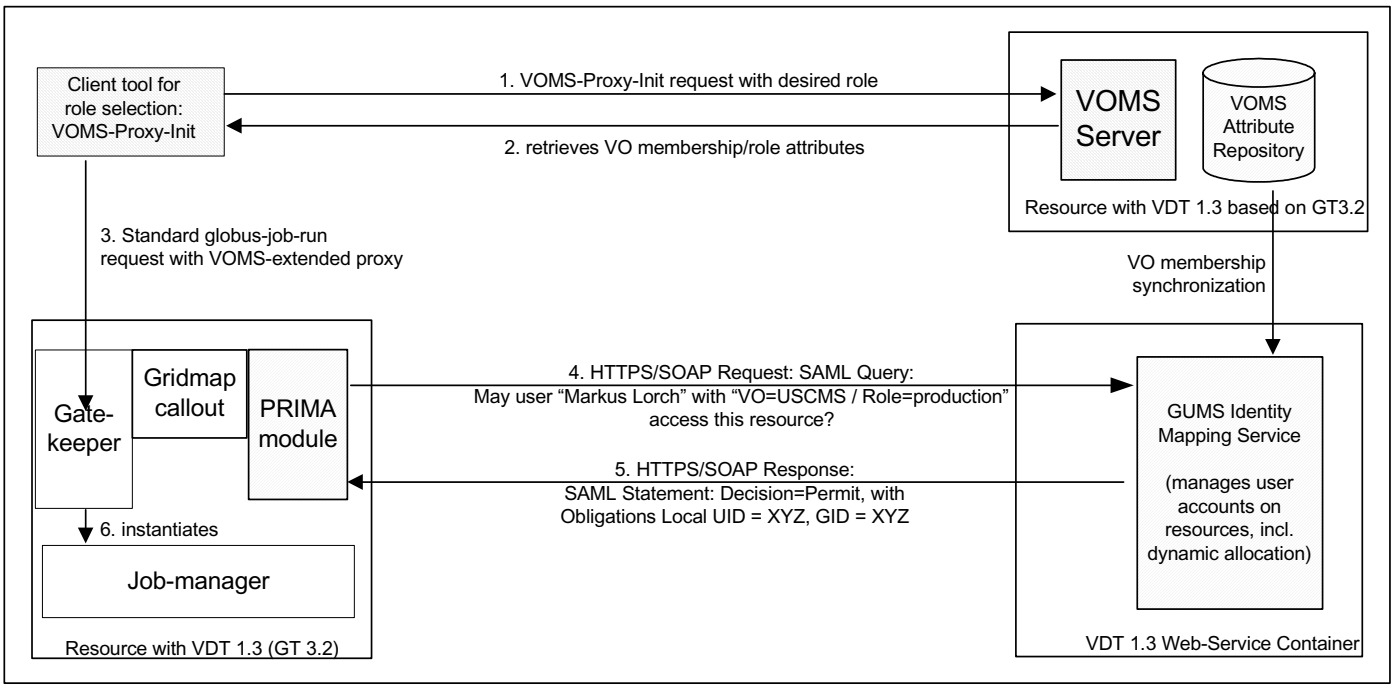

Fig. 1. Basic Privilege Project authorization architecture for compute resources

II. COMPUTE AUTHORIZATION ARCHITECTURE

The infrastructure employs an attribute-based authorization model where privileges are effectively granted to users by assigning attribute values, such as VO-membership and roles within a specific VO, to their identity. The users can select from the set of attributes available to them and present the selected attributes to a Grid resource when access to a specific Grid service is requested. The approach of providing the appropriate attributes with a request to the resource is referred to as attribute-push and, in contrast to more traditionally used attribute-pull, enables the user to control what attributes are presented when an authorization decision is computed. The Grid resource will evaluate the presented attributes and render them against applicable policies. This process will yield the appropriate access rights for the particular access which then have to be enforced on the Grid resource providing the service.

The security infrastructure components that were available at the beginning of this project provided isolated services for aspects of the desired architecture. These components had evolved in different organizations to solve specific authorization problems in the computing resources maintained by their organization. Integrating and enhancing these components was an important goal and a difficult challenge in this project. The components, described briefly below, are the Globus gatekeeper and GridFTP server [2], the GUMS Grid User Management System [3], the VOMS VO Membership Service [4], the SAZ Site Authorization Service [5], and dCache, a storage service [6]. A key integrating component, the PRIMA module [1], employed concepts and implementations that were developed as part of the PRIMA project.

The PRIMA Module is a dynamically loadable authorization module that is present on every Grid resource. It replaces the existing grid-map file functionality on Globus resources. The PRIMA Module interfaces with Globus services like the Globus gatekeeper and GridFTP server through an authorization callout originally developed in previous work [1] and later incorporated by the Globus team as an integral part of the pre-ogsa Grid security architecture.

The Grid User Management System (GUMS) has been extended as part of the project to an online identity mapping service. GUMS maps a Grid entity to a local username at the requested resource based on the entity's X.500 name and provided attributes. Thus, GUMS provides for the site-centralized, site-consistent allocation of local user accounts. A variety of allocation algorithms are possible including the dynamic allocation from a pool of user accounts, the mapping to role-specific shared accounts, and the mapping of individual (statically allocated) accounts.

The Virtual Organization Membership Service (VOMS) maintains a database of members and member roles for a specific VO. VOMS can issue user attributes that certify this information for authorization purposes. In previous Grid infrastructures earlier versions of VOMS were used as a database from which a batch process would generate static grid-map files and provision them to Grid resources. The capability of VOMS to act as an online attribute authority and issue VO attributes has not been leveraged in earlier Grids.

Figure 1 shows the overall architecture for compute resources. The numbered lines show the sequence of actions for submitting and authorizing a request for service at a compute resource.

The user requests (Step 1) the credentials that will authorize the service request to be submitted. The user can choose to either generate a generic short-lived proxy certificate using the standard "grid-proxy-init" or to use the VOMS tool "voms-proxy-init". With "voms-proxy-init" the user can customize the proxy certificate with a chosen VO and role attribute. "voms-proxy-init" will contact a VOMS server to request a trusted attribute statement from the VOMS server of the desired VO.

In response (Step 2) the VOMS server provides a signed X.509 Attribute Certificate with the requested VO-membership and desired role information if the requesting user is indeed a member of the VO and holds the desired role. The attribute information is encoded as a Fully Qualified Attribute Name (FQAN) as described in [7]. At the client side the attribute certificate is in turn embedded in the proxy credential as a certificate extension. If a user wishes to change roles a new proxy certificate must be created with the new VOMS attribute (possibly from a different VOMS server) embedded. The user can have multiple proxy certificates at any time and select the appropriate proxy certificate via an environment variable. Integration of this mechanism with credential storage solutions such as MyProxy [8] is also supported. The use of standard proxy certificates without VOMS attributes continues to be supported and provides for the backwards compatibility and ease-of-use for users with only a single (default) VO-membership and role.

Once a service request is received (Step 3) by the gatekeeper (or GridFTP server) the grid-map callout dynamically locates and invokes the PRIMA module based on information from a simple configuration file. The information provided by the grid-map callout includes the authenticated user's distinguished name (DN) as well as the security context established during authentication (which in turn holds all certificates, including the VOMS issued attribute certificate, if provided). The PRIMA module, implemented as a set of C and C++ libraries, extracts and validates the VOMS attribute certificate and parses the attribute information. To be able to verify the validity of the attribute certificates the PRIMA module must have the service certificates of all trusted VOMS servers available. However, due to the heritage of GUMS (see below) verification of attribute certificates may not be required if identity mapping is performed by GUMS.

The identity-mapping and authorization service is contacted (Step 4) by the PRIMA module. Using the Security Assertion Markup Language (SAML) [9], an Authorization Decision Query is formulated which contains the authenticated user DN (SAML Subject) as well as the VOMS FQAN attribute (SAML Subject Evidence). This query is sent to GUMS via SOAP over an HTTPS connection. The PRIMA – GUMS communication is based on the interface described in [11] and is also used by the OGSA based components of the Globus Toolkit for authorization decision queries. This commonality improves interoperability and provides for a smooth transition from the current pre-OGSA services used in the Open Science Grid to OGSA based components in the future.

GUMS, after processing the request will respond with either a SAML Authorization Decision Statement or an extended version called an Obligated Authorization Decision Statement (Step 5). The obligated statement is discussed in more detail in Section IV and enables the GUMS service to augment the basic permit/deny/indeterminate decision supported by SAML with additional decision qualifications, such as the local user account to be used for this access. The PRIMA module, upon receipt of the decision statement, will return the appropriate local user account name to the Globus gatekeeper or GridFTP server via the grid-map callout interface. The gatekeeper or GridFTP server can then continue processing the request in the same way as if a local grid-map file had been used to retrieve the local user account name.

The authorization process described above provides opportunities for backward compatibility with the existing Globus mechanisms and possible future extensions in two ways. First, GUMS maintains an SQL database with information about which user DNs are members of which VOs and their associated roles. This redundant storage of membership information is not necessary to reach a secure authorization decision in the described attribute push model of the privilege project infrastructure. However it enables GUMS to also operate in a legacy mode in which it creates standard grid-map files for distribution to Grid resources that do not have the PRIMA module available and thus cannot perform an online query. Furthermore it allows for deployments where the PRIMA module is configured to skip attribute verification and thus alleviates the need to maintain trusted attribute authority certificates (VOMS server certificates) on the Grid resources. GUMS knows all possible members of a VO and a forged attribute cannot be used to achieve member access.

Second, the currently implemented obligation formats also enable GUMS to provide information on what UNIX system group accounts (primary and supplemental groups) are to be set for the requested access (Fig. 1 depicts primary group information to be specified by GUMS). If this feature is used the PRIMA module cannot simply return the group names to the Grid service as the grid-map callout interface cannot accommodate these parameters. Instead the PRIMA module will take responsibility for setting up the appropriate execution environment by changing supplemental groups, primary group and user account of the current process before returning control to the Grid service. This allows dynamic changes in the mapping of operating system group accounts independent of the information specified e.g., in /etc/passwd and /etc/group which empowers the authorization system to take full advantage of the operating system group security semantics based on the VO roles selected by the user. This group mapping functionality may be used in future deployments of the Open Science Grid.

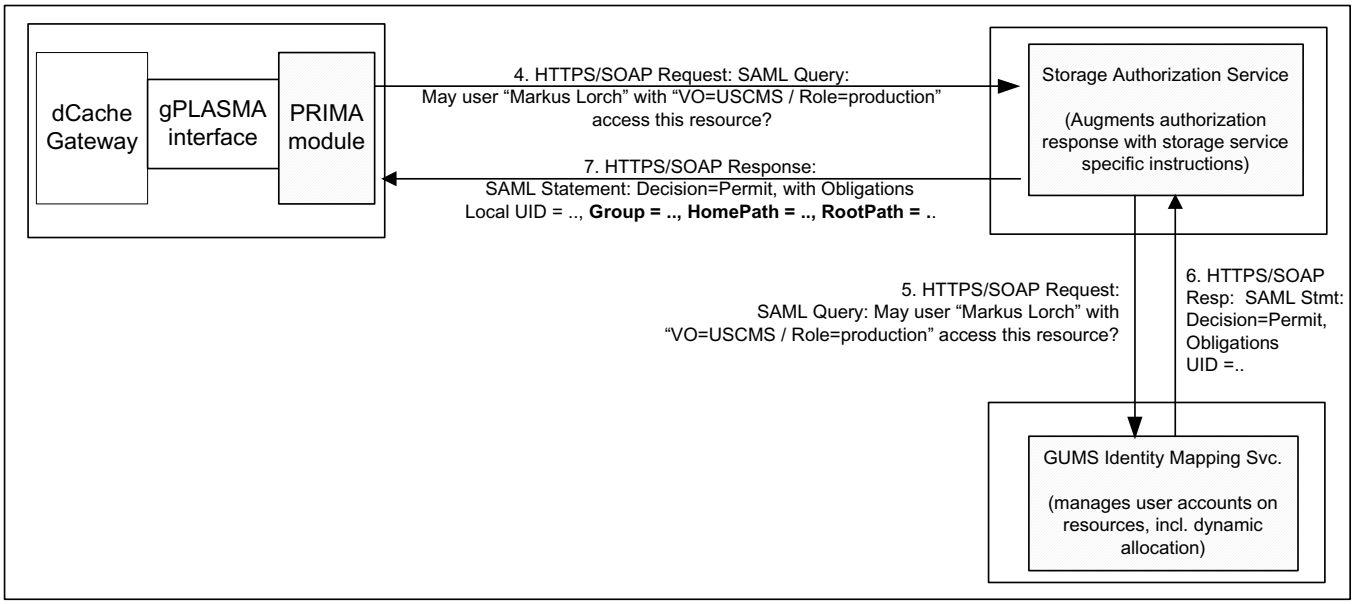

Fig. 2 Storage Authorization Architecture and Sequence

A closely related authorization component that may be combined with this infrastructure is the Site Authorization Service (SAZ) [5]. There exists a SAZ client module that is similar to the PRIMA module in that is invoked by the Grid service and queries a service, the SAZ service, for an authorization decision. SAZ enforces the site-specific access control rules/policies such as specifying prohibited users, checking for revoked certificates, and validating the user's certificate path. The SAZ service developed at FNAL is currently relying on a proprietary protocol for communication. It is planned that future versions will implement the same SOAP/SAML protocol and authorization interface used in the PRIMA-GUMS communication.

The performance of the new framework is well within acceptable ranges. The overhead introduced by the call-out to the PRIMA module and the local communication with GUMS is minimal and does not add a significant delay to the authorization procedure. For example tests during the deployment on the Open Science Grid testbed showed delays on the order of 0.5 seconds/request. The site-centralized GUMS server can sustain a reasonable number of requests (e.g., on an older machine 50 requests/second were possible). Load-balancing and redundancy for GUMS servers is possible through the underlying Tomcat 5 web service container.

The first deployment of the Open Science Grid (OSG) utilizes the basic Privilege Project components discussed in this section. The following section will elaborate on the use of the Privilege Components in conjunction with storage services which will be included in the next version of the OSG middleware package.

III. Storage Authorization Architecture

Data storage resources share with compute resources the requirement to map the Grid DN of a user to a locally known user account name. Mapping files, similar to those traditionally used for compute resources, are also used on storage resources and thus the same identity mapping functionality can be applied. In fact, for data access on compute resources via GridFTP the infrastructure described in section II works in exactly the same way as for compute access.

More complex storage systems, such as dCache [6], may require additional information before such systems can grant access. Examples include the root path and home path for this access and if the access has to be restricted to read-only operations. Traditionally this information is made available to the storage system via a mapping file. It was one of the goals of the Privilege Project to also integrate such storage resources into the authorization infrastructure and reduce the managerial overhead of maintaining such mapping files for storage resources as well. The resulting architecture, using a Storage Authorization Service that acts as a proxy between the storage system and GUMS and provides the additional storage specific information, is pictured in Fig. 2 and described below.

Steps 1 to 3, the selection of VO and role through the creation of a proxy certificate with VOMS-issued attributes and the user initiated service request to the storage resource are identical with those in Fig. 1 and not pictured in Fig. 2. There is no semantic difference to the user between a compute and a storage service access.

When a storage service access request is received by the dCache storage system the gPLASMA interface calls the PRIMA module with user the credentials provided to the storage system during authentication. The PRIMA module extracts and optionally verifies presented attributes and formulates a SAML Authorization Decision Query. This query is sent to the Storage Authorization Service (Step 4), which exposes the same SAML authorization port type that

was added to GUMS. The Storage Authorization Service, upon receipt of such a query, passes on this query to the site's GUMS server (Step 5). If the GUMS server returns a positive response (with an obligation that specifies a local user name, Step 6) then the Storage Authorization Service queries its local policy file for additional authorization decision qualifications that must be provided to the storage resource with the decision. If GUMS returns a negative response then no further processing on by the Storage Authorization Service is required. Finally the augmented SAML response, including additional obligations for the storage resource, is provided back to the PRIMA module of the dCache service (Step 7).

IV. Obligated Authorization Decisions

Obligations are a set of instructions provided with an authorization decision statement or response. These instructions may be targeted at the Policy Enforcement Point (e.g., the gatekeeper and operating system of a compute resource) and may be used to describe how a requested service, if allowed, should be confined and monitored during its execution. In this context obligations may also be used to convey additional instructions for how to treat a service request that is not authorized.

Obligations in authorization decision statements can be used to address the mismatch in the level of detail between the authorization request and the applicable policies. This mismatch is one of the more subtle issues frequently encountered when authorization decisions are made by an external Policy Decision Point (i.e., the Storage Authorization Service). Policy decision points are application independent and are unable to understand or extrapolate the implications of an arbitrary resource request such as a request to instantiate a user-provided service. The applicable policies together with provided attributes are likely to specify in detail what a user provided service is allowed to do. But a simple permit/deny decision from the decision point cannot convey this level of detail. A positive authorization decision response thus needs to be augmented with additional decision qualifications that instruct the enforcement point how exactly the requested action should be permitted and if additional constraints should be applied. For example, the list of fine-grained access rights that specifies to what extent the user provided service is allowed to access other services and resources of the hosting environment can be provided this way. Alternatively a reference to an existing execution environment or user account that is already preconfigured with the appropriate access rights can be provided. If the PEP cannot fulfill the obligations then it should not allow the access to proceed. The enforcement point, upon receiving a positive response from the decision point, instantiates or selects an appropriate execution environment configured with the access rights as specified in the obligations, and starts and monitors the execution of the requested service in this environment.

The Privilege Project has extended the SAML Authorization Decision Statement to create an "Obligated Authorization Decision Statement" that holds at least one obligation following the XACML obligation format. Each XACMLObligation element specifies if it is to be fulfilled on a permit or deny response. Fulfillment of a XACMLObligation translates to the application of the attribute assignment that the obligation statement conveys. A set of attribute assignments can be provided with a single obligation. The XACMLObligation format does not describe the semantics of the attributes that are assigned and is completely independent of the application.

The use of the XACML Obligation format allows the seamless integration with XACML policies and policy decision functions. An XACML Obligation can simply be embedded in the applicable XACML Policy and will automatically be included in the authorization decision statement that is conveyed to the enforcement point. Thus service specific policies can be written that provision service specific authorization information (such as the rootPath obligations for storage elements explained in Section III) while maintaining a service agnostic PDP implementation. Furthermore, the use of XACML Obligations will enable the Privilege Project to transition seamlessly to the new XACML over SAML authorization message format in the future. [12]

V. Shaping Execution Environments

The Workspace Service is an alternative solution to the procurement and management of local user accounts implemented in GUMS. This section presents a brief introduction to the Workspace Service and discusses how this service can be used together with the other Privilege Project authorization components. The Workspace Service has just recently been released as a technology preview component of the Globus Toolkit 4.

A workspace defines a "sandbox" or an execution environment, an isolated user environment reflecting the user's level of privilege, sharing, software preferences, and other factors relevant to a controlled execution environment. Such workspaces can be dynamically created and managed via a Grid service interface to adjust their lifetime, the shape of the sandbox, or access and management policies for various Grid entities. We experiment with a variety of workspace implementations [13]. The current production version simply provides access to an existing environment (a configured platform) by generating a Unix accounts for a Grid client [14, 15]. Another implementation takes advantage of superior isolation and enforcement characteristics of virtual machines [16] to provide a more flexible implementation. The current production version has been adopted by the EGEE [17] project and is discussed here.

The infrastructure allowing for creation and management of workspaces is composed of a GT4 factory service that allows

an authorized Grid client to create individual accounts or groups of accounts, and a workspace service that allows an authorized Grid client to manage individual account properties, such as account access policy or time to live (TTL). These concepts are represented in WSRF and implemented using the GT4 implementation of WSRF: e.g., account properties (such as TTL or policy information) are implemented as WSRF resource properties and available for inspection and modification in standard ways.

As with GUMS, accounts can be allocated based on an X.509 proxy credential (including a VOMS proxy). If the proxy contains attribute information, this information is processed in the policy information point (PIP) which then makes the attributes available for authorization (i.e., processing by the policy decision point (PDP)) and for customization of sharing and access policy. In the current implementation accounts are customized by, for example modifying Unix group settings on an account. The default PDP implementation is based on simple attribute-based and name-based access control lists (ACLs) with permit overrides rule. Account management actions (such as extending TTL or changing access policies on an account) are also authorized in similar ways. The Grid entity that created an account can for example create a policy enabling another Grid entity to access or even manage an account – however, carrying out such operation will be subject to resource owner's policies.

The back-end implementation of workspace creation and management can rely on different mechanisms according to site policies and preferences. At this point, our implementation supports two kinds of such "back-ends": (1) true dynamic creation (i.e., using the Unix "useradd" command), and (2) leasing implementation, based on mapping the Grid entity requesting an account to an existing account belonging to a pool of accounts created for this purpose earlier. Our implementation of the latter strategy is based on the LCMAPS extension of the gridmapdir patch. We augmented this implementation by developing methods for configurable and secure termination of account leases including account quarantines and configurable account cleaning procedures.

The workspace service works in conjunction with a resource management and job startup service on a given platform by exporting interface allowing a service to query for the association of a Grid entity with any account or a specific account on a given site. This interface can be used for example by a GRAM authorization callout. This interface has been extended to use the same SAML authorization port type used by the Privilege Project and is able to respond to queries with Obligated Authorization Decision Statements that specify the appropriate local user account (i.e., the user's workspace) to be used for a requested access. As a result users can leverage the management interface provided by the Workspace Service and the Privilege Project infrastructure can later query the Workspace Service for the appropriate local user account.

VI. PRIMA Authorization Service

The PRIMA Authorization Service is currently an experimental component of the Privilege Project infrastructure. This service is discussed here to illustrate the additional expressiveness and flexibility that a general purpose authorization service will add to the Privilege Project infrastructure.

The PRIMA Authorization Service is based on the XACML PDP discussed in [1], it renders authorization requests against policies in eXtensible Access Control Markup Language. The service exposes the same SAML authorization interface used by other Privilege Project components. The general processing logic is as follows: The first step after the receipt of a SAML authorization decision query is to gather information to create a request to the XACML policy engine. In the XACML model the component that implements this functionality is referred to as the (request) context manager. In the PRIMA Authorization Service this context manager is part of the authorization service itself. It analyzes the request and, based on the requested action to be authorized, first queries a GUMS service for a local user account that would apply to the requested access. The response from GUMS (particularly the attribute assignment in a returned Obligated Authorization Decision Statement) is then embedded in the request to the policy engine together with other attributes of the user that may have been provided in the original SAML request to the PRIMA Authorization Service. In the existing implementation GUMS has a veto capability in that a negative response to a GUMS query for a local user account would automatically trigger a negative response from the PRIMA Authorization Service.

The next step is to leverage the policy engine to render the request against the set of XACML policies that are available to this PDP. The policy engine is based on the SunXACML implementation of the XACML standard. The policy engine will evaluate the applicability of the request to the available policies. If a policy matches then the rules of this policy will be evaluated to render a positive or negative decision. Each XACML policy can include a set of obligation statements that are bound to the outcome of the decision (i.e., apply when decision = permit or apply when decision = deny). All applicable obligation statements will be included in the decision that the policy engine returns. This result (a XACML response) is then converted into a SAML Obligated Authorization Decision statement (the XACML decision is mapped to the SAML decision and the XACML obligation statements are copied into the XACML obligation field). The final SAML statement is then returned to the Grid service that originally requested the authorization decision. In summary the PRIMA Authorization Service can be viewed as consisting of a SAML to XACML translator, a Context Manager to retrieve the GUMS local user account attributes and a general purpose XACML policy decision point.

We have successfully shown that it can be used to provide

the same functionality as the Storage Authorization Service by simply writing appropriate policies that contain the service specific instructions in the XACML policies. No changes to the PRIMA Authorization Service are necessary to support new obligations with service specific information. The only two entities that must be able to understand the obligations are the policy administrator and the enforcement point (e.g., the storage service). The content of obligations and their meaning must not be understood by the PDP implementation which enables the use of the general purpose PRIMA authorization service for many, if not all, services. Merely the policies have to be written with the specific service requirements in mind and have to reflect obligations that can be understood by the services themselves. A single PRIMA Authorization Service can be configured with policies for many different services.

The use of a rule-based policy, such as those encoded in XACML, has administrative and scalability advantages over the use of configuration files that basically follow an access control matrix approach such as the configuration file of the Storage Authorization Service. The access control lists in Grid environments are typically very large but also very sparsely populated and difficult to maintain. For example most users of the VO USCMS can read the VO data input files but only a few have permission to modify these file. In an access control matrix every user would have to be listed with its access control rights, where as in a rule-based policy only the view "exceptions" with additional rights have to be listed.

VII. Outlook and Conclusion

Development of the Privilege Project software components will continue and the software will be improved and extended based on experiences from the integration into the OSG middleware and the large-scale deployment in the OSG at the time of this writing. For example one of the improvements will be to free the PRIMA module from the task of extracting and validating attribute certificates on the Grid resources. This task will be dealt with at the site-centralized authorization services (e.g., to GUMS). The SAML authorization request will include the complete certificate chain that was used to authenticate the user to the Grid resource. The authorization service will then extract and validate the attributes. This change will improve scalability of the overall infrastructure as the distributed Grid resources no longer need to know about trusted attribute authorities (e.g., VOMS servers).

Acknowledgment

We would like to acknowledge the contributions of other members of the Privilege Project, the Virtual Data Toolkit team and the Open Science Grid integration activity.

References

[1] M. Lorch, D. Kafura "The PRIMA Grid Authorization System", Int. Journal of Grid Computing (2004) 2: pp. 279–298.

[2] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture" in "The Grid, Blueprint for a Future Computing Infrastructure", I. Foster, and C. Kesselman, Editors, Morgan Kaufmann, San Francisco, 1999, pp. 259-278

[3] G, Carcassi "The Grid User Management System". http://grid.racf.bnl.gov/GUMS

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

[5] 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

[6] M. Ernst et al., "Managed Data Storage and Data Access Services for Data Grids", In proc. of CHEP 2004, Interlaken, Switzerland

[7] M. Thompson, V. Welch, ***M. Lorch***, R. Lepro, D. Chadwick, "Attributes used in OGSA Authorization", http://forge.ggf.org/projects/ogsa-authz/document/draft-OGSA-authorization-attributes.pdf

[8] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy", *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, 2001.

[9] P. Hallam-Baker, E. Maler, et al, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), Oasis Standard, November 5th, 2002

[10] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[11] V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, L. Pearlman, "Use of SAML for OGSA Authoriztion", drafts available from https://forge.gridforum.org/projects/ogsa-authz/document/draft-ogsa-authz-saml-feb1-05.doc/en/1

[12] XACML – SAML Profile. See http://www.oasis-open.org/committees/download.php/10525/XACML-2.0-SAML-PROFILE-CD-02.zip

[13] Keahey, K., K. Doering, and I. Foster. *From Sandbox to Playground: Dynamic Virtual Environments in the Grid*. in 5th International Workshop in Grid Computing. 2004.

[14] Keahey, K., M. Ripeanu, and K. Doering. *Dynamic Creation and Management of Runtime Environments in the Grid*. in *Workshop on Designing and Building Web Services*. 2003. Chicago, IL.

[15] *Workspace Management Service*: http://www.mcs.anl.gov/workspace/

[16] Keahey, K., I. Foster, T. Freeman, X. Zhang, and D. Galron, *Virtual Workspaces in the Grid.* Europar 2005

[17] *EGEE Global Security Architecture* , EU Deliverable DJRA3.1, https://edms.cern.ch/file/487004/1.1/