DRAFT COPY (to be published in *Journal of Grid Computing*)

ROAM: An Authorization Manager for Grids

J. R. Burruss[1], T. W. Fredian[2], M.R. Thompson[3]
[1]*General Atomics, burruss@fusion.gat.com*
[2]*Massachusetts Institute of Technology, twf@psfc.mit.edu*
[3]*Lawrence Berkeley National Laboratory, mrthompson@lbl.gov*

Abstract

The Resource Oriented Authorization Manager (ROAM) was created to provide a simple but flexible authorization system for the FusionGrid computational grid. ROAM builds on and extends previous community efforts by both responding to access authorization requests and by providing a Web interface for resource management. ROAM works with the Globus Resource Allocation Manager (GRAM), and is general enough to be used by other virtual organizations that use Globus middleware or X.509/TLS authentication schemes to secure a grid of distributed resources. In addition to describing ROAM, this paper discusses the basic design parameters of a grid authorization system and the reasons for the choices made in the ROAM design.

1 Introduction

The National Fusion Collaboratory [1] is a virtual organization consisting of researchers from the three major U.S. plasma physics research centers and collaborators from other universities and labs. The goal of fusion research is to develop a clean, sustainable energy source, and the goal of the Collaboratory is to advance fusion research by providing new tools to scientists. The collaboratory has created a computational grid, FusionGrid, consisting of computational services and data repositories. FusionGrid is designed to simplify the use of large, complex legacy fusion codes, unify data access, and provide administration and security processes to enable researchers from different organizations to work together in a single virtual organization. This is important as fusion research is and will continue to be a highly collaborative effort. Moving forward, the next-generation International Thermonuclear Experimental Reactor (ITER) fusion device will be located in France but must be accessible to collaborators around the globe. FusionGrid services are geographically and administratively dispersed; it is expected that the lessons learned in managing these services can be applied to future international projects such as ITER.

The Collaboratory started work on FusionGrid in 2001. The first FusionGrid service came online in 2002; software from the Globus Toolkit™ [2] was used for remote job submission and secure data transfer. The fusion data management system MDSplus [3] was updated to use the Globus Security Infrastructure for secure data transfer and authentication of users. These early efforts were a success, having a positive impact on fusion science capabilities [4].

However, FusionGrid, like many other grids of the time, found the Globus Resource Allocation Manager (GRAM) "grid mapfiles" [5] to be difficult to manage and too inflexible for the planned expansion of resources. The basic grid mapfile scheme required each individual host to maintain a separate grid mapfile to map FusionGrid certificates to local accounts. Having mapped certificates to local accounts, it was left to each individual resource administrator to implement both grid-wide and local authorization policies. Experience demonstrated that it was hard to maintain coherence with mapfiles distributed across multiple machines at multiple sites. Furthermore, the use of grid mapfiles precluded variable account mapping because of the lack of wildcard specification and the need to edit each individual mapfile by hand. This is especially problematic when the same user is mapped to different local accounts on one machine depending on the resource being used, or when a resource is spread across several machines. Also, the "all or nothing" access granted by the grid mapfile was not sufficient for the access control needs of computer servers, which wanted to restrict use of what executables a grid user could run. What was needed was a flexible system to provide grid-wide authorization and account mapping.

2 Related Work

Initial efforts to improve authorization in FusionGrid focused on the Akenti authorization system [6]. Akenti is an authorization system designed to handle distributed resources controlled by multiple stakeholders. FusionGrid developers worked with GRAM developers to define and include an authorization callout in the job manager so that GRAM could consult Akenti for authorization information. This callout to Akenti allowed fine grain access control based on the code to be run and the job parameters the user requested. It supplemented the coarse grain

admission control provided by the grid mapfile and allowed sites to closely control what grid users could do. Akenti was used with success with the first FusionGrid service. However, the Akenti implementation of authorization information as distributed, digitally signed documents made some of the desired access management operations difficult. In particular it is not easy to list all the users and resources of the virtual organization, or to check on all the outstanding authorizations for a given resource.

FusionGrid developers looked at the Community Authorization Server (CAS) [7] and the Virtual Organization Membership Service (VOMS) [8]. CAS implements the push model of authorization where the user must first contact an authorization server to get some credentials and then present the credentials to the resource provider. These credentials are in addition to the X.509 [9] credential used to establish identity and may consist of attributes such as roles and account id or grants of specific permissions for resources. The resource provider must then verify the credentials and often do additional local authorization checks. This model was undesirable because FusionGrid architects wanted to keep the authorization path as simple as possible, both for scientists and for developers. VOMS can operate in either a push or a pull mode, but the pull mode functions by periodically updating a site gridmap file [10] which is too coarse-grained for FusionGrid, having the property that a user will get mapped to only one set of permissions per site rather than per resource. Furthermore, the design of VOMS focuses on defining users and their attributes or permissions. The FusionGrid administrators wanted a system that defined the resources of a virtual organization as well as the users and their permissions. It was felt that having a unified listing of resources and users would simplify the assignment of rights to users.

3 Goals

It was decided to create a custom authorization system capable of maintaining a coherent authorization scheme over the distributed resources of FusionGrid. Taking their cue from the Akenti model, developers determined that the new system must allow for multiple stakeholders to control access to resources. For example, in the case of code running at computer centers, both the site administrators and the owner of the code needed to be able to restrict or allow access. It was also determined that the desired system must be capable of variable account mapping.

Developers also considered the "soft" requirements of stakeholders. Developers wanted as little disturbance as possible to their existing codes, and administrators wanted an intuitive interface to view, add and remove access to their resources. Thus, developers opted for a simple access control decision interface that could be easily used by both people and codes and that would bring together the distributed authorization information into a coherent picture. To assuage the concerns of stakeholders that "outside" computers would control their systems, the system would be consultative, not controlling; in other words, resources would consult the system for authorization information as opposed to having outside systems control resources directly. With these goals in mind, FusionGrid staff set about creating what would be known as the Resource Oriented Authorization Manager (ROAM).

4 Roam Model

4.1 Information Model

The heart of ROAM is its information model. The ROAM information model is designed to represent FusionGrid authorization in a coherent way. It consists of a framework of *resources*, *permissions*, *users*, and *authorizations*.

The focus of the ROAM information model is the *resource*. A resource is typically a grid service, but it can also be an entire site, such as Massachusetts Institute of Technology (MIT) or General Atomics. In general, a resource is anything that needs to have its own authorization list. If you have to fill out a form to get permission to use something—for example to access a network, to read or write to a database, or to execute a code—then that thing can be modeled as a resource in ROAM.

A *user* is any uniquely identified consumer of resources. This is typically a human with a FusionGrid certificate, but it can also be a program or service with an identifying certificate. Resource administrators and other stakeholders are users, as are FusionGrid researchers and engineers.

A *permission* is a type of usage for a resource. Each of these permission types represents a way in which a resource is used. Examples are "access" for a site, "read" or "write" for data, and "execute" for a code.

An *authorization* is a grant of a specific permission for a particular user on a specified resource. ROAM authorizations are binary in nature; you either have an authorization or you do not have an authorization. An authorization indicates that user X has permission Y on resource Z. Authorizations may include contexts, which can be used to supplement the simple binary authorization with additional information. On FusionGrid, context is currently only used to specify the local username or group name under which an action should be performed.

It is worth noting that the ROAM information model is not intended for role-based access control (RBAC) [11]. This is a deliberate design choice made by FusionGrid architects as it was determined that roles—while

useful for large organizations—are an extra complexity not required to model grid-wide authorization policy for FusionGrid.

4.2 Data Model

For the most part, the implementation of the ROAM information model is straightforward. Essentially, the ROAM data model is a Lampson protection matrix [12] implemented in an object-relational database, with one table each for resources, users, permissions, and authorizations (Fig. 1). However, in order to keep the list of permissions for a given resource to a minimum, a table of resource-permissions was added to the data model. This table indicates the list of valid permissions for a given resource. Thus, even if new resources introduce into the database a large volume of specialized permissions, each individual resource will have associated with it only those permissions applicable to that resource.

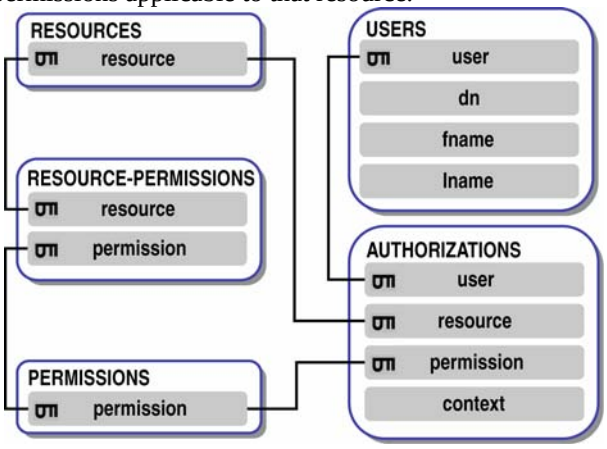

Fig. 1: The ROAM data model consists of resources, users, permissions, authorizations, and a list of valid permissions for each resource.

This very simple model is flexible enough to satisfy current FusionGrid resource stakeholders. As the number of users or projects within FusionGrid grows it may be useful to add the concept of groups (or experiments) and possibly roles to the framework; in such a case, permissions could be granted on the basis of group membership and/or roles, which is a standard way to handle access for large numbers of users or a large number of resources.

4.3 Queries

Concomitant to the simple data model are the simple queries needed to answer authorization questions. ROAM authorizations are binary in nature and the authorization queries reflect this characteristic. A ROAM query example: "does user X have permission Y on resource Z?" Although this is very simple, it can be used with real life authorization policies and understood by users, administrators, and other stakeholders.

On FusionGrid, a typical authorization policy is no more complicated than "To use a code hosted at a site, a user must be authorized by the authors of the code to use that code, and must be authorized by the administrators of the site to use computing resources at that site". The following figure (Fig. 2) illustrates the query process for a specific example of this typical two-rule policy. In this example, a service S_1 needs to confirm that a user U_0 has both the access permission (represented as P_0) to site S_0 (the site where the code S_1 is located), and that the user has execute permission (represented as P_1) for the code S_1. This is accomplished with two sequential queries to ROAM; assuming the answers (A_0 and A_1) are both "yes", the user is authorized to use the service because the user has satisfied both the author of the code and the site administrator. If either stakeholder in this example—i.e., the author of the code or the site administrator—has not authorized the user, then permission is denied.

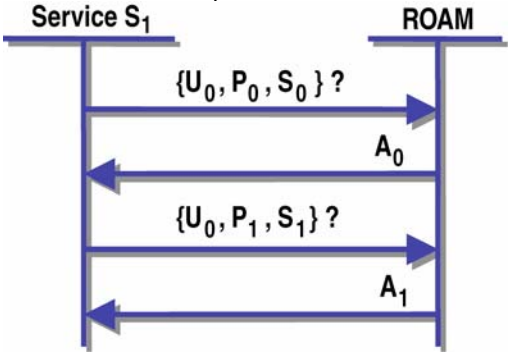

Fig. 2: A service S_1 checks to see if user U_0 is authorized.

As noted previously, any authorization can also be defined with additional context information. This auxiliary information is implemented as a string in the ROAM database; it is left to the resource to interpret the meaning of this string. Clients performing an authorization query can optionally include a parameter asking for context information. In this case, the context string will be returned instead of a simple affirmative response.

5 Architecture

ROAM consists of two tiers (Fig. 3). The bottom tier (or back-end) is a relational database containing all the users, resources and authorizations for the various FusionGrid resources. The top tier (or front-end) is a Web interface which accepts secure Hypertext Transfer Protocol (HTTPS) [13] connections from users, stakeholders and resource providers. The front-end provides a management interface and an authorization decision interface.

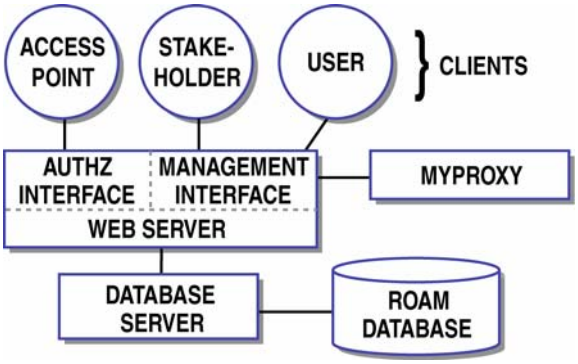

Fig. 3: Clients interact with ROAM through the web interface.

Stakeholders and other users access the management interface through a Web browser. Each connecting user is identified by an X.509 certificate. The connections are made via HTTPS [14] with client-side authentication requested. Users without a certificate in their Web browser are redirected to a login page where, assuming they enter a valid FusionGrid username and password, they are authenticated and a proxy certificate is received from a MyProxy [15] server. Programs that mediate access to the FusionGrid resources also call the authorization interface via the HTTPS protocol. The interface responds with, simply, a "yes" or a "no" to indicate the existence of an authorization for the specified user, resource, and permission plus any contextual information that is asked for.

ROAM avoids the push model of authorization (where the client must first contact an authorization server to get some credentials and then present them to the resource provider). Instead, clients connect to resources as they would normally, using an X.509 proxy credential from MyProxy to authenticate. The resource then consults ROAM to see if the connecting user is authorized (Fig. 4). In this way the authorization path is completely transparent to the user.

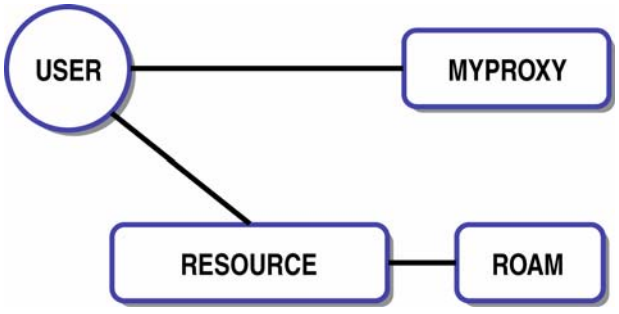

Fig. 4: After signing on to the grid through MyProxy, a user accesses a resource normally; the resource queries ROAM directly.

Note that it is the resource itself that decides what combination of permissions are needed, if any external resources are involved, if contextual information is needed and how that contextual information is to be used. This has the effect of allowing the resource stakeholders to set the authorization policy for the resource at the resource access point. This permits creativity in security policy by resource developers, particularly with the use of the contextual information. While this local policy implementation decreases the grid-wide visibility of authorization policy, developers felt that the flexibility and additional assurance that it provided to the local resource providers was worth the tradeoff. In the initial FusionGrid implementation of ROAM, local policy rules have been expressed both in Bourne shell scripting language and in the Tree Data Interface (TDI) MDSplus expression language. TDI is an interpreted expression evaluation language that is familiar to the fusion researchers. See Section 8.3 for more discussion of TDI versus other policy language choices.

6 Implementation

FusionGrid developers implemented the ROAM back end in a PostgreSQL [16] database. The front end was built using web Common Gateway Interface (CGI) [17] scripts written in PHP [18] running on an Apache web server [19]. The GridPort portal and toolkit [20] was evaluated for this interface, but after consideration it was decided that the learning time and complexity of running a complete portal was more effort than was necessary for initial FusionGrid requirements. Both the database and the web server run on the same Linux host, thus eliminating the need to secure a network connection between them. Each of the technologies selected by FusionGrid developers to implement the ROAM back end and front end are mature, open source and freely available.

6.1 Management Interface

The ROAM access management pages are configured to authenticate the client using Secure Sockets Layer (SSL) [21]. If the client browser has a valid certificate (i.e. one signed by the FusionGrid Certificate Authority) the client is identified by the distinguished name in the certificate. If no valid certificates are loaded into the client browser, ROAM prompts the user for a username and password, then uses the entered username and password to retrieve a proxy certificate from the FusionGrid MyProxy server. If the username and password are valid then the user is identified by the distinguished name in the retrieved proxy certificate.

Once the user has authenticated using a certificate, the user can, through the ROAM Web pages, request authorization to use a grid resource, view a log of their resource queries, create a new resource, or manage an existing resource (Fig. 5). A stakeholder for a resource can see all the existing authorizations, (i.e. user, permission, context) for a resource and can add or delete authorizations.

Authorization actions such as requests, grants, and revocations result in email notification to concerned parties, though in the case of a grant or revocation email to the user is optional. In practice, resource administrators typically wait for email notification of a pending request, then click a link embedded in the email to process the request.

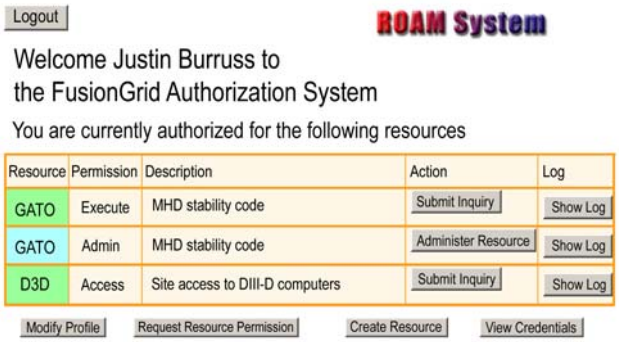

Fig. 5: Both user and administrative actions can be executed through the ROAM web interface.

6.2 Authorization Decision Interface

The authorization decision interface consists of HTTPS queries containing the user's Distinguished Name (DN), the name of the resource, the permission that is needed, and optionally a request for any context associated with the permission. The reply consists of a "yes" or "no" answer and a context string if one exists and was asked for.

FusionGrid resources use GRAM for remote job submission and GridFTP for data transfer. By configuring the GRAM gatekeeper and GridFTP to use a gridmap callout, these Globus services were modified to query ROAM and use the context information for specifying the local user account. This removed the need to manage grid mapfiles to specify the mapping of credentials to local accounts. Similarly, MDSplus was modified to enable the MDSplus expression evaluator to provide mapping from user certificates to local user accounts using this same auxiliary information. For example, MDSplus can query ROAM to determine if a user is authorized to connect to an MDSplus data server, then use the context information of the resource permission to specify the local user and group list.

6.3 Robustness Considerations

To avoid what would otherwise be a single point of failure in FusionGrid, a secondary ROAM server was deployed. It is used in much the same way as a secondary Domain Name System (DNS) [22] server: clients first try to query the primary ROAM server, then failover to the secondary ROAM server if the primary is down. The primary ROAM server is installed at MIT while the secondary ROAM server is installed at Lawrence Berkeley National Laboratory (LBNL). The secondary ROAM server operates in a read-only mode: it will respond to authorization decision requests, but only supports the management interface for read requests. The secondary server's database is updated once per day from the primary server; this interval is arbitrary but is currently sufficient to meet FusionGrid needs.

7 Status

ROAM was deployed for FusionGrid in 2004; the first FusionGrid service to use ROAM was the GATO [23] service which provides access to a well known plasma stability code. The GATO service was configured to query ROAM through a call to curl in a Bourne shell script. Subsequently, MDSplus databases at DIII-D in San Diego and Alcator C-Mod at MIT were configured to check ROAM for user authorization. Most recently, ROAM was put into use by the TRANSP FusionGrid service which runs another tokamak plasma analysis and simulation code. To date, ROAM has been very reliable. The secondary ROAM server has already been used on a few occasions when the primary server was down. However, there were two network partition events arising from offline routers that did cause problems: in these cases the timeouts were exceedingly long and administrators were forced to manually swap the primary and secondary ROAM server entries for the duration of the partition.

To date ROAM has processed authorization queries with peak loads of 854 queries per hour, 62 queries per minute, and 16 queries per second under test conditions. This performance under test conditions greatly exceeds the day-to-day needs of FusionGrid and is arguably very good considering that ROAM runs on an older Pentium II Linux host with only 128 MB of RAM. Normal rates of access peak at around 5-10 queries per minute for data access and considerably less for starting codes and accessing sites.

8 Discussion

The decision to design and implement a custom authorization service for the FusionGrid came after two years of experience using the basic Globus grid mapfiles and the Akenti authorization system. The creation of gatekeeper and job manager authorization callouts—motivated by community dissatisfaction with the lack of flexibility and maintainability of grid mapfiles—enabled this decision. Once the authorization decision was separated from the access enforcement, it became possible to easily experiment with authorization services. The design of ROAM considered the following dimensions: centralization versus distributed policy, push or pull model of credential presentation, balance of authorization policy between access point and authorization point, and customized functionality versus general purpose solutions.

8.1 Centralized versus Distributed Policy

The FusionGrid was launched using the Akenti Authorization system, which was designed to support distributed authorization policy. A policy for a single resource would consist of several signed documents, each representing contributions from the multiple stakeholders for a resource. The pieces would be gathered up at the authorization time, validated, and combined to produce and access decision.

It turned out that in the FusionGrid, as in most virtual organizations, the policy distribution was not a useful feature. Since virtual organizations, almost by definition, support some secure central facilities that all the authorized members of the virtual organization can reference, this was the obvious place to store commonly managed access policy. Policy distribution is more useful in peer-to-peer and ad-hoc environments that do not have central facilities.

In an environment such as FusionGrid that supports secure central servers, and provided the load is not too high, centralized authorization has several advantages. First, it is easy to provide an overview of all authorization information to the virtual organization administrators when information is centralized. Second, it is simpler and more efficient to find the information at access time. Third, it is easier to ensure security of centralized information than to secure data distributed across multiple locations.

8.2 Credential Presentation

There are two basic approaches for a user to present their credentials to a resource provider. The simplest from the user's point of view is to provide proof of identity and then expect the authorization server to discover any other credentials that might be needed for authorization. This approach is called the *pull model* and corresponds to legacy access control systems. The other model is the *push model*, where the user gathers up any credentials such as identity, attributes or authorization assertions and presents them to the resource provider. This model is similar to capability based systems where the user presents an authorization token that the resource provider needs only to validate. Arguably, the push model is easier for the resource provider to implement since users do the work of gathering permissions, leaving resources to simply validate tokens. On the other hand, the multiple steps required complicate resource usage from the point of view of the user.

8.3 Balance of Authorization Policy

Some authorization services, especially ambitious ones, aspire to define all policy statements and make all the decisions. General purpose policy representation is often complex in order to deal with stakeholder-defined attributes, hierarchical roles, the context the user is coming from, the conditions at the access site and more. Complex policies evaluated by the authorization point require the passing of lots of information from the user and the service provider to the authorization service. The understanding of how the access decision is made is centralized in the authorization server and may leave the resource provider uncertain as to who has access to their resource. At the other extreme, the access enforcement point makes the decision in-line, based only on the authenticated identity of the user. Transparency is lost when policy is written into source code which is then compiled; this makes it impossible to quickly assemble an overall view of grid-wide policy.

The FusionGrid stakeholders decided on a balance that left much of the control at the access enforcement point, i.e., in code that they were writing. ROAM provides global naming for users, resources, per-resource permissions, and provides global visibility of context information. But exactly how this information is used is left up to rules written by the resource providers. While this could lead to complicated and difficult to understand code, the FusionGrid stakeholders have a convenient expression evaluation language known as TDI that they have been using for years to make MDSplus queries. TDI contains the sequential, repetition, and decision structures fundamental to any structured programming language. Since it is a programming language, TDI empowers resource developers to develop, if needed, complicated authorization rules. These TDI functions are stored as text with the resource provider and interpreted at execution time by MDSplus libraries that are available on the FusionGrid resource sites. These rules can be made visible locally and used as templates for other resources. Because TDI is interpreted, no restarts or recompilations are required to pick up changes.

Some consideration was given to Security Assertion Markup Language (SAML) [24] for authorization assertions and eXtensible Access Control Markup Language (XACML) [25] for policy rules. These languages were not selected as it was determined that they were more general than was required, were still immature and thus subject to changes by third parties, and would have led to a dependency on third party eXtensible Markup Language (XML) [26] parsers, which were also immature and subject to change. In contrast, TDI is very mature, having been deployed in production environments for over 15 years, and having over that time maintained backwards compatibility.

The idea of specifying access policy as executable rules has a precedent in the PolicyMaker [27] work. Here, filters could be written in an interpreted language such as sed or awk that would enforce rules based on the user credentials and any local environmental conditions to determine if a requested action was allowed. In the

future, the Semantic Web languages Resource Description Framework (RDF) [28] and Owl [29] will enable XML languages to express executable rules (and the evaluation engines to execute the rules) for general policy enforcement.

The decision to allow the stakeholders to write the policy rules for their resources, coupled with the recommendation to do so in publicly accessible TDI functions, allows the stakeholders to feel that they are safely in control of who has access to their code, and helps to encourage people to make their codes and resources available through the FusionGrid.

8.4 Customized Versus General Solution

Finally, a major influence behind the design and implementation of ROAM was the realization that FusionGrid did not require sophisticated general purpose authorization or virtual organization membership management. In fact, using unnecessarily complicated tools discouraged participation in the FusionGrid by potential resource providers. Thus, ROAM was written using familiar tools and providing no more features than were necessary.

9 Future Work

FusionGrid developers are currently testing a higher-level timeout-to-failover mechanism to repair the network partition timeout problem. The goal is to implement a network partition failover mechanism that will not require administrator intervention.

Another suggestion integrates ROAM and the FusionGrid Monitoring system (FGM) [30]. Currently, FGM keeps a record of general-purpose monitoring information from FusionGrid resources, and ROAM keeps a record of all authorization queries. If the two systems are merged, then authorization attempts could be automatically posted from ROAM to FGM. As it now stands, developers implement their own authorization posting mechanism for each new resource, thus leading to duplication of effort.

Currently, the authorization queries from the resources are secured only by the ROAM server certificate. One idea under consideration is to increase security by requiring a client certificate from the resource managers. This requirement would prevent unauthorized entities from ascertaining individual accessibility to grid resources.

In the longer term, it is expected that ROAM could be used by other virtual organizations that need to maintain coherent authorization schemes for resources distributed across multiple administratively separated domains. In particular, either ROAM or a ROAM-like system would be appropriate for international collaborations such as ITER.

Acknowledgments

The authors wish to thank our National Fusion Collaboratory Project colleagues for their help with ROAM. This work was funded by the SciDAC project, U.S. Department of Energy at General Atomics under contract number DE-FG02-01ER25455 and cooperative agreement number DE-FC02-04ER54698, and by the Director, Office of Science, Office of Advanced Science, Mathematical, Information and Computation Sciences of the U.S. Department of Energy at Lawrence Berkeley National Laboratory under contract number DE-AC03-76SF00098.

References

1. D.P. Schissel, et al., “Building the U.S. National Fusion Grid: Results from the National Fusion Collaboratory Project”, in *Fusion Engin. Design*, Vol. 71, pp. 245-250, 2004.

2. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. “A Security Architecture for Computational Grids”, in *Proc. 5th ACM Conf. on Computer and Communications Security*, San Francisco, California, United States, November 2-5, 1998, pp. 83-92.

3. T.W. Fredian, J.A. Stillerman, “MDSplus: Current Developments and Future Directions”, in *Fusion Engin. Design*, Vol. 60, p. 229, 2002.

4. J.R. Burruss, et al., “Remote Computing using the National Fusion Grid”, in *Fusion Engin. Design*, Vol. 71, pp. 251-255, 2004.

5. K. Czajkowski, et al., “A Resource Management Architecture for Metacomputing Systems”, in *Proc. 4th Workshop on Job Scheduling Strategies for Parallel Processing in Conjunction with IPPS/SPDP '98*, Orlando, Florida, March 30, 1998, p. 62.

6. M. Thompson, A. Essiari, S. Mudumbai, “Certificate-based Authorization Policy in a PKI Environment”, in *ACM Transactions on Information and System Security (TISSEC)*, Vol. 6, No. 4, pp. 566-588, 2003.

7. L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, “A Community Authorization Service for Group Collaboration”, in *Proc. IEEE 3rd Intl. Workshop on Policies for Distributed Systems and Networks*, Monterey, California, June 5-7, 2002.

8. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell 'Agnello, A. Frohner, A. Gianoli, K.L Lorentey, and F. Spataro, “VOMS: an Authorization System for Virtual Organizations”, in *Proc. 1st European Across Grids Conf.*, Santiago de Compostela, Spain, February 13-14, 2003.

9. R. Housley, et al., “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, RFC 2459, http://www.ietf.org/rfc/rfc3280.txt, 2002.

10. D. Britton, P. Clarke, J. Coles, D. Colling, A. Doyle, S.M. Fisher, A.C. Irving, J. Jensen, A. McNab and D. Newbold, "A Grid for Particle Physics – from testbed to production", Tech Report, University of Glasgow, GLAS-PPE/2004-05, http//:ppewww.ph.gla.ac.uk/preprints/2004/05/2004-05.doc

11. E.C. Lupu, et al., "A Policy Based Role Framework for Access Control", in *Proc. 1st ACM Workshop on Role-Based Access Control (RBAC '95)*, Gaithersburg, Maryland, November 30 — December 2, 1995.

12. B.W. Lampson, "Protection", in *Proc. 5th Princeton Symp. on Information Sciences and Systems*, March 1971, reprinted in Operating Systems Review, Vol. 8, No. 1, pp. 18- 24, 1974.

13. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, http://www.ietf.org/rfc/rfc2616.txt, 1999

14. E. Rescorla, "HTTP over TLS", RFC 2818, http://www.ietf.org/rfc/rfc2818.txt, 2000

15. J. Novotny, S. Tuecke, V. Welch, "An Online Credential Repository for the Grid: MyProxy", in *Proc. 10th IEEE Intl. Symp. on High Performance Distributed Computing (HPDC-10 2001)*, San Francisco, California, August 7-9, 2001, pp. 104-111.

16. Momjian, Bruce, PostgreSQL: Introduction and Concepts, Addison-Wesley, 2000

17. The Common Gateway Interface, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html, accessed December 5, 2005

18. M. Achour, et al., PHP Manual, http://www.php.net/manual/en/, November 25, 2005, accessed December 5, 2005

19. Laurie, Ben and Laurie, Peter, Apache: The Definitive Guide, 3rd edition, O'Reilly, 2002

20. M. Thomas, et al., "The GridPort Toolkit Architecture for Building Grid Portals", *Proc. 10th IEEE Intl. Symp. on High Perf. Dist. Comp.*, Aug 2001.

21. A. O. Freier, "SSL Protocol V. 3.0", http://wp.netscape.com/eng/ssl3/ssl-toc.html, March 1996, accessed December 5, 2005

22. M. Mockapetris, "Domain Names – Concepts and Facilities", RFC 1034, http://www.ietf.org/rfc/rfc1034.txt, 1987

23. L. C. Bernard, et al., "GATO: An MHD stability code for axisymmetric plasmas with internal separatrices", *Computer Physics Communications*, Vol. 24, 377, 1981

24. P. Mishra et al., "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)", http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf, December 2001, accessed December 5, 2005

25. S. Godik, et al., "OASIS eXtensible Access Control Markup Language (XACML)", http://lists.oasis-open.org/archives/wsia/200205/pdf00001.pdf, May 2002, accessed December 5, 2005

26. F. Yergeau, et al., "Extensible Markup Language (XML) 1.0 (Third Edition)", http://www.w3.org/TR/2004/REC-xml-20040204/, February 2004, accessed December 5, 2005

27. M. Blaze, J. Feigenbaurm, and J. Lacey "Decentralized Trust Management", *Proc. IEEE CS Symp. on Security and Privacy*, Oakland, California, May 6-8, 1996, pp. 164-173.

28. "RDF Primer," in F. Manola, E Miller (eds.). W3C Recommendation, February 10, 2004, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, accessed April 3, 2006

29. "Owl Web Ontology Language Reference," in M. Dean, G. Schrieber (eds.), W3C Recommendation, February 10, 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, accessed April 3, 2006

30. S. Flanagan, et al., "A General Purpose Data Analysis Monitoring System with Case Studies from the National Fusion Grid and the DIII–D MDSplus Between Pulse Analysis System", *Fusion Engin. Design*, Vol. 71, pp. 263-267, 2004.