

Integrating Legacy Authorization Systems into the Grid: A Case Study Leveraging AzMan and ADAM

Weide Zhang, David Del Vecchio, Glenn Wasson and Marty Humphrey

Department of Computer Science, University of Virginia, Charlottesville, VA USA 22904
{ wz6y, dad3e, gsw2c, humphrey }@cs.virginia.edu

Abstract. While much of the Grid security community has focused on developing new authorization systems, the real challenge is often integrating legacy authorization systems with Grid software. The existing authorization system might not understand Grid authentication, might not scale to Grid-level usage, might not be able to understand the operations that are requested to be authorized, and might require an inordinate amount of "glue code" to integrate the native language of the legacy authorization system with the Grid software. In this paper, we discuss several challenges and the resulting successful mechanisms for integrating the Globus Toolkit and WSRF.NET with AzMan, a role-based authorization system that ships with Windows Server 2003. We leverage the OGSA GGF Authorization Interface and our own SAML implementation so that the enterprise can retain their existing AzMan mechanism while resulting in new, scalable mechanisms for Grid authorization.

1 Introduction

Constructing a Grid requires the integration of a potentially large number of new and existing software mechanisms and policies into a reliable, collaborative infrastructure. One of the biggest challenges in this integration is security, particularly authorization: after verifying that the person is whom they say they are (authentication), is the person allowed to perform the requested action? While the Grid community has created a number of excellent authorization systems such as CAS [1], VOMS [2], PERMIS [3], and AKENTI [4], these systems are generally assumed to be *installed* at or around the same time as the Grid software such as the Globus Toolkit [5] or WSRF.NET [6]. However, in many cases it is unrealistic to assume that the adoption of Grid technology means the abandonment of authorization mechanisms already in place. Hence the real challenge is often to integrate a *legacy* authorization system with Grid software. The legacy authorization system might be closely tied to an existing authentication system and might not be able to understand new authentication assertions/tokens. It might be designed to work with a relatively few number of users/objects and might not scale to the size of the Grid being considered. Perhaps it does not understand the requested actions and therefore cannot represent and make decisions about the proposed actions, such as "launch remote job" or "read a file via GridFTP". It is not clear how

difficult it would be to get the Grid software to implement the protocol of the legacy authorization system.

This paper describes the integration challenges, approach, and lessons learned as we attempted to integrate the role-based access control (RBAC) system that is shipped with Microsoft Windows Server 2003 (Authorization Manager, or "AzMan" [7]) with the Globus Toolkit v4 and WSRF.NET as part of the University of Virginia Campus Grid [8]. A key to our solution is that this paper reports one of the first implementations of the GGF OGSA Authorization Interface [9]. Our integration not only allows an enterprise to continue using its AzMan installation as it installs Grid technology, we have found that the Grid management is further enhanced over the state of the art in improved support for dynamically modifying authorization policies, maintaining a consistent view of site-wide policies, and reducing the cost of policy management.

2 Legacy Authorization System: Overview of AzMan and ADAM

In this section, we describe AzMan, the legacy authorization system in our case study. AzMan (Section 2.1) is a general-purpose, role-based authorization architecture on Windows platforms. Section 2.2 describes ADAM, a lightweight Windows service for directory-enabled applications, which we use in combination with AzMan.

2.1 AzMan

In traditional access control mechanisms based on Access Control Lists (ACL), users are directly mapped to resource permissions using a list of authorized users for each target resource. In many situations, ACL-based systems do not scale well, in that if a new user "Fred" is introduced into the system, all of the objects which he should have access to must have their ACLs changed. Role-Based Access Control (RBAC) [10] adds a *role* layer between users and permissions. For example, assume that before Fred comes along, resources have been designed to allow certain operations according to well-defined roles such as "salesperson". Then once Fred is hired, rather than changing the ACLs on all resources Fred needs access to, he just needs to be recognized as having the "salesperson" role. Since users can typically be categorized into a number of different roles, RBAC tends to be a more scalable and flexible approach.

Conceptually, the primary purpose of AzMan is to provide a "yes" or "no" answer when asked at run-time (via a Microsoft COM API) if a particular authenticated identity is allowed to perform a particular action. AzMan also provides a graphical interface and a separate API for entering and configuring identities, roles, permissions, etc. AzMan allows for the definition of any number of access control policies, the central concepts of which are *Roles* and *Permissions*. Subjects are assigned Roles, and Roles are granted or denied Permissions for certain tasks. Roles can actually be assigned to either individual subjects or groups of security principals. Such grouping can even be computed at run-time based on an Active Directory (LDAP) lookup. In addition to dynamic groups, policies themselves can also have a dynamic element. In particular, they can reference *BizRules*, which are scripts that get executed when particular Per-

missions are requested, so that run-time information like "time of day" can be used to make an authorization decision.

Typically, AzMan authorization policies are grouped into named policy sets based on the application to which they apply. Note, however, that access rights are not directly associated with specific target resources (as an ACL would be for a file in the filesystem). In fact, policies are completely resource-independent; AzMan requires authorization decision requests to identify only the subject, intended task and the name of policy set. When an application that uses AzMan is initialized, it loads the authorization policy information from a policy store. AzMan provides support for storage of authorization policies locally in XML files on the AzMan server, or remotely in Active Directory (or ADAM, next section).

2.2 Active Directory Application Mode (ADAM)

Active Directory Application Mode (ADAM) is a relatively new capability in Active Directory that addresses certain deployment scenarios of directory-enabled applications [11]. In contrast to Active Directory, ADAM can be used for storing information that is not globally interesting. One example usage in an authorization decision-making context involves storing the names of policies that apply to a particular resource on a per-resource basis. ADAM is also valuable for those situations in which a particular application must store personalization data for users who are authenticated by Active Directory. Storing this personalization data in Active Directory would sometimes require schema changes to the user class in Active Directory; ADAM can be used as an alternative.

3 Leveraging AzMan and ADAM for Grid Authorization

The challenge then, is to use our legacy authorization system (AzMan and ADAM) for Grid authorization, thereby providing minimal disruption to the enterprise when attempting to deploy a Grid. In our architecture, resource information is distributed among a collection of ADAM servers based on the resource's DNS name. This allows the owner/administrator of the resources in each sub-domain to configure his or her ADAM server independently, including the authorization policy that should be applied, thereby providing the domain autonomy that is so vital to the Grid. The ADAM servers are organized hierarchically: queries to a parent ADAM server will be forwarded through to the appropriate child sub-domain.

Policy management is divided into two parts: RBAC policy management and resource-to-policy mapping. RBAC policy management includes defining roles and role permissions while resource-to-policy mapping involves defining which RBAC policies apply to a resource or resource group (the latter being the responsibility of resource owners). Out of the box, AzMan only supports assigning roles to subjects identified by Windows security tokens, which each have a unique Security Identifier (SID) [7]. In multi-organization Grids, however, X.509 Distinguished Names (DNs) are usually used to identify subjects. So for AzMan to work with Grid identities, we setup a map-

ping between X.509 DNs and unique custom-defined SIDs. Since this mapping could be used by several authorization servers, we made DN-to-SID lookups possible via a Web service interface. This subject-mapping Web service uses a flat XML file to store the mappings, which is fine for relatively small numbers of subjects; a more sophisticated storage mechanism (like a relational database or XML database) could easily be substituted. With this subject mapping service in place, roles in AzMan can easily be assigned to custom SIDs as desired. All domains within a Virtual Organization (VO) share AzMan policy set names and the policy definitions they contain so that consistent authorization decisions can be made across the VO.

A typical authorization workflow is as follows (see Figure 1). A client sends a

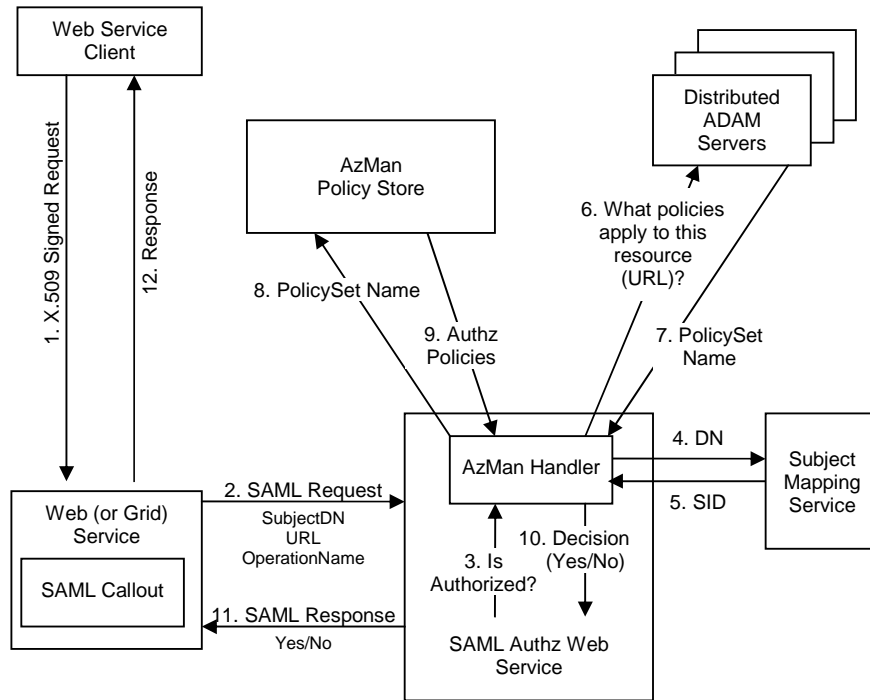


Figure 1. Authorization Workflow using AzMan and ADAM.

signed request message to a Web service (step 1) which defers authorization decisions to an authorization callout library. The SAML callout library contacts a SAML Authorization Web service (2), sending it the client's DN, the URL of the target service/resource, and the name of the requested operation. The SAML Authz Web service is relatively generic and relies on the AzMan Handler library (3) to provide the glue code that understands how to query AzMan for Grid authorization decisions. The first action the AzMan Handler takes is to retrieve from the Subject Mapping Service, the SID that corresponds to the subject DN (4, 5). Next, the distributed hierarchy of

ADAM servers will be queried (6) to determine which RBAC policy set (7) applies to the requested resource. At this point, the AzMan engine will be invoked to make an authorization decision based on the subject's SID, the returned policy set name and the desired operation (or task). The AzMan engine will take care of retrieving the indicated set of policies from its policy store (8, 9), identifying the roles assigned to a subject SID and checking the permissions assigned to those roles. AzMan's authorization decision (10) is then relayed back through the SAML Authz Web service to the original Web service (11) which proceeds to execute the desired operation (if authorized) and return the results to the client (12).

A key component of our approach is the OGSA Authorization Interface [9]. SAML [12] is used as a format for requesting and expressing authorization assertions to a SAML Authorization Web service. A standard XML message format for SAML requests, assertions and responses is provided. SAML defines a request/response protocol in which the request contains {subject, resource, action and supporting credentials} and the response contains either authorization assertions about the subject with respect to the resource or a yes/no decision. While, in principle, the Web service could use authorization assertions to make its own authorization decision, we have found that Web services typically can't or don't want to make such decisions. In addition, returning a yes/no answer allows the Web service owner to be separate from the entity that sets access control policy.

A Java-based SAML Callout library is distributed as part of the latest version of the Globus Toolkit as one option for enforcing access control in Grid Services. We provide a similar (and interoperable) callout library for Web services on the .NET platform. The SAML Authz Web service shown in Figure 1 is a .NET Web service that is relatively generic with regard to how it actually makes its authorization decisions. It can be configured (via a configuration file, or even at run-time) to use one or more authorization handler libraries. In this paper we've considered AzMan as our legacy authorization system, but any other authorization system could also be substituted into our architecture with appropriate authorization handler glue code. Since the SAML Authz Web service is interoperable with both .NET and Java and callout libraries, AzMan (or another authorization system) can be used to make authorization decisions for Web services running on just about any platform. Thus we accommodate the heterogeneity that is present in many organizations and multi-organization Grids.

4 Evaluation

We now assess our architecture and implementation as described in Section 3.

System Scalability: Cost of Administering Authorization Policies. Modifying authorization policy in this system requires two steps: updating the RBAC policies in the policy store, and updating the mapping of those policies to specific resources. In our system, authorization policy modification is inexpensive. The RBAC policies can be updated through AzMan's COM interface or the Microsoft Management Console (MMC), a graphical management interface. In order to update the policy that applies to a specific resource, the resource provider or authorized user changes the resource/application mapping stored in distributed ADAM servers. The worst case cost of policy modification (that affects every user, every role and every resource) in our infrastructure is $U+P+R$, where U is the number of users, P is the number of permissions and R is the number resources, in the VO. The vast majority of policy changes, however, only require modifications along one of these three dimensions.

Resource Scalability: Cost of Handling Dynamic Grids. In large Grid environments, many resources are not static, but instead change dynamically. In our system, resource groups and policies that govern those groups are maintained separately (in ADAM and AzMan, respectively). Resource owners/administrators can freely update information about resources in particular domains and sub-domains by changing the information stored in distributed, hierarchical ADAM servers. The resource-independent policies in the AzMan policy store will be unaffected by these resource changes with resource-to-policy mappings inspected at run-time. In contrast to this architecture, other authorization systems such as Permis [3], Akenti [4], and CAS [1] all store policies that contain resource information directly. In these systems, there is no upper bound on how many policies will be affected when a resource changes (for example, has its trust domain revoked).

Membership Scalability: Cost of Dynamic User Membership. Resources are not the only dynamic element in large-scale Grids, users change as well. Consequently, our RBAC policies stored in AzMan do not mention users directly. Instead, our strategy is to define globally unique identifiers (SIDs) that map to X.509 DNs. RBAC policies are specified in terms of roles assigned to SIDs. Changes to the set of users in a organization only requires a change to the subject DN-to-SID mapping table. Changes in role assignments are similarly localized and the policy definitions themselves are not affected by them. Some other authorization infrastructures store User Attribute Certificates or directly embed membership information into authorization policies, yielding a less scalable approach.

Resource Provider and VO Independence. Our authorization infrastructure separates the responsibility for mapping roles to permissions from the responsibility for applying policies to resources. This allows resource providers independent control over which policies apply to their resources while allowing VO administrators to define policies that (potentially) apply to all resources in the VO. This separation allows resource providers to react quickly (and separately) to change policies for their resources in the event of security breaches or other malicious activity taking place in the VO. Similarly, it allows VO administrators to effectively remove resources from the

VO (by removing permissions from roles) if those resources are deemed to be acting inappropriately.

Trust Management. There are three aspects of trust that are relevant to any deployment of our system: trust of users, trust of the services involved in authorization, and trust of the authorization policies. Trust of users refers to the mechanism by which the system can trust the user and authenticate the user's identity. In our UVA Campus Grid, the UVA CA acts as an Identity Authority that services are configured to trust. Any service must therefore verify that the user's certificate is signed by the UVA CA, to establish trust. Trust of the authorization services refers to the mechanism by which the distributed ADAM servers (each of which lies in a different administrative domain) can trust the Authorization Service. In the UVA Campus Grid, the (SAML) Authorization Web service impersonates a user account to access distributed ADAM servers. This user account is set beforehand in the ADAM servers and is granted the "read" privilege so that the Authorization Service will be able to read policy names applied to particular resources. The Subject Mapping Service must also be trusted by the Authorization Service with X.509 signed messages between these two services assumed. Trust of authorization policies refers to how the Authorization Service can trust the policies that come from the AzMan service. In the UVA Campus Grid, the Authorization Service is simply configured to trust the AzMan service (and hence the policies it provides). An improvement on this would be to have each policy provider sign the policies, allowing different Authorization Services to determine if different policies come from sources they trust.

Consistent View and Flexibility of the AuthZ Policy. In our infrastructure, the authorization decision point dynamically retrieves the name of policies that apply to a resource from that resource's ADAM server. Members of a Virtual Organization (VO) will have a globally consistent view of the named policy sets and their constituent policy definitions since the AzMan policy store is accessible across the VO. In some authorization infrastructures, policies are either defined locally or signed policy certificates (in Akenti [4]) are specified for a target resource so that authorization decisions must be made locally. Without a consistent global view of policies, it becomes more difficult to make certain kinds of policy changes, for example check and modify some local policies from previously denying access to allowing a certain user access to the target resource. Also, there is also no guarantee users who possesses VO-wide administrator roles can be authorized for access across multiple administrative domains.

5 Summary

In this paper, we presented both grid authorization requirements and a new authorization infrastructure based on AzMan, authorization mechanisms included with Windows Server 2003. Our integration not only allows an enterprise to continue using its AzMan installation as it deploys Grid technology, but we have found that Grid au-

thorization management is further enhanced over the state of the art through improved support for dynamically modifying authorization policies, maintaining a consistent view of VO policies, and reducing the cost of policy management. Future work will focus on analyzing the scalability and reliability issues the hierarchical and distributed ADAM servers introduce as well as better understanding the performance overhead of our authorization architecture.

References

- [1]. I. Foster, C. Kesselman, L. Pearlman, S. Tuecke, and V. Welch “The Community Authorization Service: Status and Future”, in *Proceedings of Computing in High Energy Physics 03(CHEP '03),2003*
- [2]. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, K. Lörentey and F. Spataro, “From gridmap-file to VOMS: managing authorization in a Grid environment.” in *Future Generation. Comput. Syst.* 21, 4, 549-558(2005).
- [3]. D. Chadwick, A. Otenko, “The PERMIS X.509 Role Based Privilege Management Infrastructure”, in *Future Generation Comp. Syst.* 19(2), 277-289(2003)
- [4]. M. Thompson, S. Mudumbai, A. Essiari, W. Chin, “Authorization Policy in a PKI Environment”, 2002. [Online]. Available: <http://dsd.lbl.gov/security/Akenti/Papers/NISTPKWorkshop.pdf>
- [5]. Globus project. <http://www.globus.org>
- [6]. M. Humphrey and G. Wasson. Architectural Foundations of WSRF.NET. *International Journal of Web Services Research.* 2(2), pp. 83-97, April-June 2005.
- [7]. D. McPherson, “Role-Based Access Control for Multi-tier Applications Using Authorization Manager” [Online]. Available: <http://www.netscum.dk/technet/prodtechnolog/windowsserver2003/technologies/management/athmanwp.aspx>
- [8]. M. Humphrey and G. Wasson. The University of Virginia Campus Grid: Integrating Grid Technologies with the Campus Information Infrastructure. *2005 European Grid Conference (ECG 2005)*, Amsterdam, The Netherlands, Feb 14-16, 2005.
- [9]. V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, L. Pearlman, “Use of SAML for OGSA Authorization”, submitted in GGF OGSA Security Working Group, Aug 2003
- [10].R. Sandhu, E. Coyne, H. Feinstein, C. Youman, “Role-Based Access Control Models”, in *IEEE Computer* 29(2), 38-47 (1996)
- [11].Microsoft Corporation. "Introduction to Windows Server 2003 Active Directory Application Mode". Nov 5, 2003. <http://www.microsoft.com/windowsserver2003/techinfo/overview/adam.aspx>
- [12].Security Assertion Markup Language (SAML) OASIS Technical Committee. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security