A Service Oriented Architecture for Authorization of Unknown Entities in a Grid Environment

J. RIVINGTON, R. KENT, A. AGGARWAL, P. PRENEY
Computer Science Department
University of Windsor
401 Sunset Avenue, Windsor, Ontario, N9B 3P4
CANADA
http://www.hpc.uwindsor.ca/

Abstract: - In many cases, within distributed environments, authorization manifests itself in the form of existing trust relationships. Before pervasive computing can be successfully achieved, we may have to transcend the current notion of pre-established trust. This is not conducive to a low administrative overhead, nor is it realistic in a distributed environment, where processing may occur over a large number of nodes which may be distributed geographically across different domains. This paper presents a unique architecture which provides a distributed authorization capability that allows arbitrary entities to participate in the grid, while greatly improving scalability due to lower administrative overhead. Within our architecture, the access decision is made at the individual resource sites, based on the combination of local policy and a set of accumulated points carried in the requesting entity's PKC. These points, which are derived from previous actions the entity has been involved with, will be used to represent an entity's reputation. The system is called Augmented Authorization System Using Reputation (AASUR).

Key-Words: - Authorization, Security, Grid, Web Services, OGSA, Reputation, Evidence

1 Introduction

The Open Grid Services Architecture (OGSA) provides a universally applicable and adopted framework for distributed system integration, virtualization, and management [1]. Though the security of web services has been addressed in the OGSA specifications, authorization is, at best, rudimentary. Currently, OGSA authorization relies on a static grid-map file, specific to each resource, which provides a mapping from a global ID (PKC) to a local account [2]. Local account rights are then used to access resources. While this certainly has a place in the general authorization scheme of OGSA, especially in the case of pre-established collaborations and federations, it is not sufficient for an evolving distributed computing environment, where resources may be shared with individuals who may be unknown to the resource provider. Moreover, if each new entity, which may or may not use a resource, must be manually added to the local gridmap file, the scalability of the system would become limited. In this paper we propose an architecture which collects and records actions performed by an entity throughout its lifetime in a grid. These actions are tracked through a subsystem. The actions are recorded as points, that are then used as evidence. When combined with site specific policies, this evidence is used to determine an entity's reputation, or to augment other local policies to obtain an authorization decision.

The rest of this paper is organized as follows: Section 2 briefly discusses current authorization systems. Section 3 provides a general overview of Web Services and Service Oriented Architecture. Section 4 describes our system of points, evidence, reputation, and the infrastructure necessary for AASUR. Security policies, messages, the files and formats, required by AASUR, are given in section 5. The architecture and workflows of our system are described in section 6. The last section discusses the scope of our implementation, its inherent limitations and future work.

2 Current Systems

Several systems have been proposed to accomplish authorization in distributed environments. The closest match to AASUR is the TERA (Trust-Enhanced Role Assignment) system [3]. Though TERA encapsulates ideas, similar to AASUR, it has several key differences. First, the reliability of the evidence is based on the trust of an evidence provider. This implies that evidence may come from a third-party source and be potentially less reliable. Our system maintains a globally consistent representation of an entity's past, which allows a resource site to make an access decision based on a local interpretation. Secondly, TERA relies on a separate reputation server to manage user reputation [3], whereas we store the actions an entity has performed in its PKC, managed by a Certificate Authority (CA). In AASUR, the reputation is calculated using local processing power to maintain scalability. Finally, whereas our architecture has been developed as a component of a Service-Oriented Architecture (SOA), TERA is an application based system.

There are other systems which also address some aspects of authorization in distributed environments. Any system which makes use of the Globus Toolkit [2, 4, 5] makes use of a gridmap file and SAML assertions [6], in varying degrees, for authorization. Such systems require accounts to be created prior to job execution. Though the possibility of a set of local generic accounts for unknown entities would solve this problem, it is undesirable from several viewpoints, including accounting. The Akenti Authorization System [7] provides a mechanism for a finer degree of specification of authorization rights from multiple owners. Shibboleth [8], a SAML-based authorization framework, provides a mechanism for combining attribute information of known entities from multiple resource sites for the purpose of an access decision. VOMS [4] and CAS [5] have been developed to provide authorization on the basis of Virtual Organization (VO) membership, which enhances scalability, but still requires the manual configuration of trust relationships and VO memberships.

3 Web Services and OGSA

Current trends are moving from monolithic architectures to compositions of small, well defined modules. This affords easier maintenance, code reuse, abstraction from unnecessary implementation details, and domain-wide access to these modules. Once properly described and maintained in a repository, using WSDL and UDDI, respectively, for discovery purposes, these publicly exposed modules can be accessed as Web Services using well known web protocols such as HTML, XML, and SOAP. These services can be used independently or composed to create more complex services.

Service Oriented Architecture (SOA) refers to a loosely coupled grouping of a set of Web Services that communicate through XML based messages, and are maintained in an enterprise wide registry for discovery [9].

4 Reputation and Evidence

If an entity is to execute on foreign resources, the system must use some globally accepted metric to determine whether or not the entity can be trusted. Typically, this metric, outside of the computing world, is the reputation of that entity.

We have developed a set of actions which, we believe, directly pertain to the act of making an authorization decision. From a high-level view, the actions (called point categories), are divided into three main sections: Illegal Actions, Legal Actions, and Others. The set of point categories, given in Tables 1 and 2, may change as our system evolves.

Although the illegal and legal categories contain the same actions, it is necessary to maintain both. The act of performing one of the legal actions is equivalent to earning one point which will be added to the associated category in the user's PKC.

Reads/Attempts	Communication attempts
Writes/Attempts	Deletes/Creates
Execution/Process Spawn	Resource allocations
System commands	Service invocations
Device Access	Various I/O Operations

Table 1 – Legal/Illegal Point Actions

Total Jobs	Successful jobs
Forcibly terminated jobs	Blacklist recommendations
Overuse of Resource	Compilation Errors
Runtime errors	Permitted job requests
Denied job requests	Buffer overflows

Table 2 – Other Point Actions

To enable assessment of reputation we use points, evidence, and reputation. Points are acquired by tracking important actions performed by a user (e.g., overuse of resources). This point base serves as evidence to apply evidentiary reasoning (including possible uncertainty associated with points) [10, 11, 12] and, subsequently, the evidence will be used by the resource site, along with local policy, to determine that user's reputation.

Although previous actions are very important, there may be additional information which a resource site may consider. Our architecture also makes use of non-numeric information. Like numeric data, the non-numeric information is transmitted in each entity's PKC. Table 3 provides examples of non-numeric evidence.

First job request	First job completed
Most recent request	Most recent job completed
Blacklist Membership	Average job time
Culture	Geographic location

Table 3 – Non-Numeric Evidence

5 Required Policies and Format

A set of policies, formats, and messages has been devised so that resource owners are able to retain ultimate control over their resources. The user's class is derived from non-numeric evidence and, along with various policies, is used to derive a Risk Factor (RF) Formula [10]. The formats defined in the system are as follows:

1. Certificate Point Format
2. Local/Global Blacklist Format
3. Message Formats
   - 3.1 Blacklist Addition Recommendation
   - 3.2 Point Addition Recommendation
   - 3.3 Point Action Notification
   - 3.4 Blacklist Search Request/Response
   - 3.5 Blacklist Update
   - 3.6 Resource Access Ticket
4. Policy Formats
   - 4.1 Risk Factor Engine Generation
   - 4.2 Access Decision Engine
   - 4.3 User Class Definition
   - 4.4 Point Earning Action Policy
   - 4.5 Local/Global Blacklist Action Policy

6 Architectural Overview

The layered design of OGSA permits straightforward integration of AASUR within existing and future systems as a web service. We assume the proper and secure use of web services, using the WS-Security suite, along with standard web service mechanisms such as SOAP and WSDL. The Certificate Authority functionality may be accessed as a SOAP service [13]. Since a rule-based approach may not be able to provide a logically correct output in degenerate cases, the enhanced authorization may not be available. In such cases the current system alone would be responsible for authorization.

The code that relates to the proposed system, and that requires deployment at all participating sites on the grid, is expected to be contained within secure, signed binaries to prevent tampering and to ensure integrity.

Figure 1 provides a graphical, step-by-step representation of the system workflow. The Figure is discussed in detail, below.

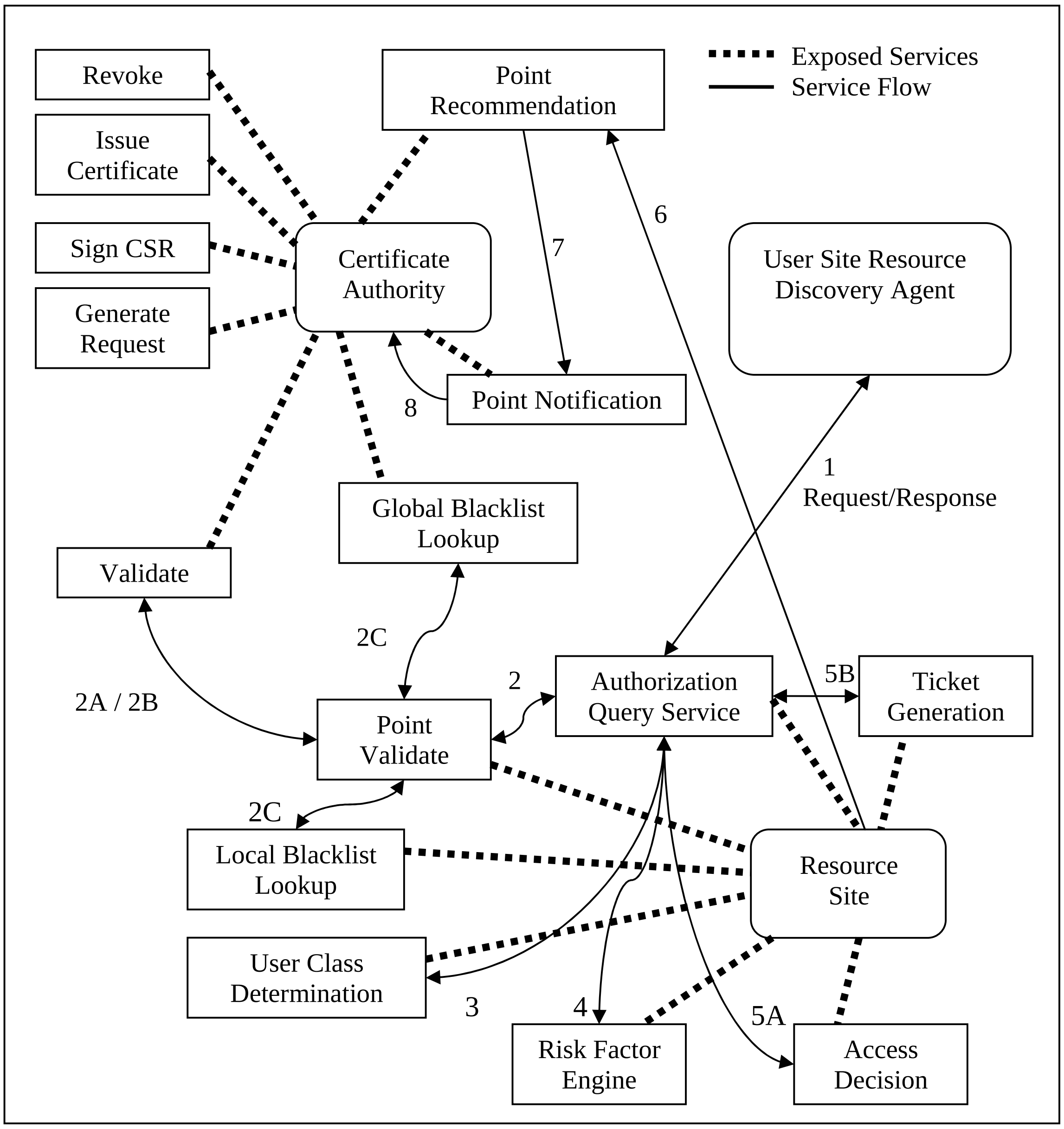

Figure 1. System workflow and Service Oriented Architecture. Note that in all cases, participating Certificate Authorities will run the service both from step 7 and step 8. However, step 7 will involve the CA of the resource site and step 8 will involve the CA of the user.

1. Resource discovery agent from the User sends a request to the **Authorization Query Service** at the Resource Site. This request includes an X.509 PKC containing past actions of the user, specified in the Certificate Point Format.
2. The point from the certificate must now be verified by the **Point Validation Service** using three checks, all applied to each point group, which is uniquely defined by the assigning resource site / point category pair. First (2A), it must be verified, that a trusted resource site recommended these points. This is done by verifying that the site has been certified by a trusted CA. Second (2B), we must ensure that the actual addition of points to the certificate has been performed and signed by a

trusted CA. Both 2A and 2B are done using the **Validate** operation of the web services CA [13]. Finally (2C), we must check to see if a particular point group was issued by a resource which has lost our trust by checking to see whether it belongs to our local blacklist, or possibly a global CA blacklist. This uses the local and global **Blacklist Lookup Service** using message 3.4 in section 5. Depending on the result, we may not accept the points added by the particular resource. Though a point group may have been added under the globally accepted policy, an individual resource site may still choose to ignore them.

3. Along with the Risk Factor, the access decision may also take into account the requesting user's class. A user's class will be determined by the **User Class Determination Service** based on additional non-point based information as defined in 4.3 in section 5. This may include attributes as in Table 3. The notion of user class is important in two places. First, the resource site may place more importance on certain categories of points depending on a user's class, for the derivation of the Risk Factor formula, as defined in 4.1 in Section 5. Secondly, the user class may be used to govern what degree of resource access will be given,, assuming that some level of access is being granted.
4. Using the **RF Engine Service**, the Risk Factor Formula is generated based on local relative weights specified in 4.1, section 5, and User Class determined in 3, a Risk Factor Formula is derived accepting all valid points as input. It will generate an absolute value of RF.
5. The **Access Decision Service** makes the final decision (5A). Essentially, it provides a yes or no answer, with possible restrictions, in the form of a ticket generated by the **Ticket Generation Service** (5B). This ticket is similar to that used in Kerberos [14]. This service uses policy 4.2 defined in section 5.

   We use the notion of a ticket for two reasons. First, within the context of the SOA, this architecture allows us to offer our authorization service, minimally, to resource discovery mechanisms. Rather than discovering appropriate resources and distributing a job simply to find out that access will no longer be given, we can determine where access will be given based on the acquisition of a ticket (3.6 in section 5). This ticket, with an explicit lifetime, will be presented upon distribution of the job to the resource site for processing. Secondly the notion of delegation is very crucial to distributed processing. This ticket would enable delegation to occur, similar to a proxy.

   Once a job has begun execution, we must monitor exactly which actions are performed by the job. We require a middleware layer which handles the detection and collection of actions performed during processing.

   It should be noted that if at any time during execution, the job performs an action that is considered globally "bad" (defined in the Global Blacklist Action Policy, 4.5 in section 5), the job will be terminated and the proper messages (3.1 in Section 5) will be sent to the parent CA of the resource site. This update will be propagated to the remaining CAs via message 3.5 in Section 5. If an action is recorded that is deemed "bad" only from the resource site viewpoint, the job may or may not continue processing, depending on 'severity', and additions will be made to the local blacklist. Note that the blacklists will exist as defined in 2.0 of Section 5.
6. Once processing has finished, the resource site will connect to its parent CA which has an instance of the **Point Recommendation Service** running locally. This service will accept message 3.3 in section 5 which contains all local actions performed. This service then maps these local actions to a set of global actions, which are consequently mapped to points (through 4.4 in section 5).

   Note that rather than requiring individual sites and users to deal with foreign Certificate Authorities, which requires complex signature chain validations, we have all entities deal only with their parent CA.

7. These points are then formatted into message 3.2 in section 5, and sent to the **Point Notification Service** running on the parent CA of the job owner.
8. Once the parent CA of the job owner receives the notification, it will perform a few simple validations, lookup the proper associated PKC, and add the points to the certificate according to format 1 in section 5. Note that the new points must be added by the parent CA of the entity which earned the points, not the CA of the resource site. This is due to the fact that only the parent CA of the certificate owner can sign the certificate.

7 Summary and Future Work

In this work we have presented a unique architecture which provides a distributed authorization capability, allowing arbitrary entities to participate in the Grid, while greatly improving scalability due to lower administrative overhead.

Future work will include the development of middleware for monitoring the actions of the entity on the grid. Failure and break-pointing of grid jobs have to be taken into account. If the proposed system is used over a long period, PKC may become very large. Techniques for limiting the size, without adversely affecting the computation of reputation must be worked out.

References:

[1] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. *The Open Grid Services Architecture, Version 1.0.* Memo, January 2005.

[2] K.Keahey, V. Welch, S. Lang, B. Liu, and S. Meder. *Fine-Grain Authorization Policies in the GRID: Design and Implementation.* Middleware Workshops 2003: 170-177.

[3] B. Bhargava and L. Lilien. Department of Computer Science. Purdue University. http://www.cs.purdue.edu/homes/bb/2004BostonIDM.html

[4] V. Alfieri, R. Cecchini, V. Ciaschini, F. Spataro, L. dell'Agnello, A. Frohner, and K. Lorentey. *From gridmap-file to VOMS: Managing Authorization in a Grid Environment.* April 2004. FGCS.

[5] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. *A Community Authorization Service for Group Collaboration.* IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[6] The Globus Security Team. *Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective.* December 2004.

[7] M. Thompson, S. Mudumbai, and A. Essiari. *Certificate-based Authorization Policy in a PKI Environment.* ACM Trans. Inf. Syst. Secur. 6(4): 566-588 (2003).

[8] V. Welch, T. Barton, K. Keahey, and F. Siebenlist. *Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration.* http://grid.ncsa.uiuc.edu/papers/gridshib-pki05-final.pdf

[9] What is SOA. http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html

[10] Jøsang A. *A logic for Uncertain Probabilities.* International Journal of Uncertainty, Fuzziness and Knowledge-Based System, Vol. 9, No. 3, June, 2001.

[11] Jøsang A. *Subjective Evidential Reasoning*. The Proceeding of the 9th International Conference in Information Processing and Management of Uncertainty in Knowledge-Based System (IPMU 2002), Annecy, France, May, 2002.

[12] Yuhui, Z. and Bharat B. *Authorization Based Evidence and Trust.* 4th International Conference, DaWaK 2002, Aix-en-Provence, France, September 4-6, 2002. Proceedings.

[13] Web Services based Certificate Authority http://soapclient.com/certService.html

[14] Kerberos: http://web.mit.edu/kerberos/www/