Policy Based Dynamic Negotiation for Grid Services Authorization

Ionut Constandache, Daniel Olmedilla, and Wolfgang Nejdl

L3S Research Center and University of Hannover, Germany
{constandache,olmedilla,nejdl}@l3s.de

Abstract. Policy-based dynamic negotiations allow more flexible authorization in complex Grid environments, and relieve both users and administrators from up front negotiations and registrations. This paper describes how such negotiations overcome current Grid authorization limitations, and how policy-based negotiation mechanisms can be easily integrated into a Grid infrastructure. Such an extension provides advanced access control and automatic credential fetching, and can be integrated and implemented in the new version 4.0 of the Globus Toolkit.

1 Introduction

Grid environments provide the middleware needed for access to distributed computing and data resources. Distinctly administrated domains form virtual organizations and share resources for data retrieval, job execution, monitoring, and data storage. Such an environment provides users with seamless access to all resources they are authorized to. In current Grid infrastructures, in order to be granted access at each domain, user's jobs have to secure and provide appropriate digital credentials for authentication and authorization. However, while authentication along with single sign-on can be provided based on client delegation of X.509 proxy certificates [21] to the job being submitted, the authorization mechanisms are still mainly identity based. Due to the large number of potential users and different certification authorities, this leads to scalability problems calling for a complementary solution to the access control mechanisms specified in the current Grid Security Infrastructure (GSI) [8].

In this paper, following up previous work described in [2], we address the limitations in current grid environments and introduce an extension to the Grid Security Infrastructure and Globus Toolkit 4.0 in which policy-based negotiation mechanisms offer the basis for overcoming these limitations. This extension includes property-based authorization mechanisms, automatic gathering of required certificates, bidirectional and iterative trust negotiation and policy based authorization, ingredients that provide advanced self-explanatory access control to grid resources.

The rest of the paper is organized as follows: in section 2 we describe the current limitations and motivate our approach. Section 3 provides a brief overview of policy based trust negotiation and further section 4 introduces this mechanism to Grid illustrating its benefits. Our proposed architecture along with a description of our implementation is given in section 5, while section 6 presents related approaches to Grid authorization. Finally, section 7 concludes our presentation and describes further work.

2 Motivation

A group of scientists at the Engineering Department of a well known University needs to obtain some simulation data regarding oceanic water waves in order to develop signaling instruments for tsunami hazards avoidance. Fortunately, the University and the Navy Institute have an agreement, allowing University members to utilize any of the Institute scientific instruments, as long as it is not already in use and the University has not exceeded its 40 monthly allocated hours using Institute resources. Both the University and the Navy Institute support Globus Toolkit 4.0 [7] providing the middleware the scientists need to collect the required data from a Wave Tank available at the Institute site.

Alice, the leader of our group of scientists, prepares a job and submits it to the University HPC Center Linux Cluster via the University Grid Portal. Together with the job, Alice delegates to the Linux Cluster an X.509 Proxy Certificate [17] which can further be used by the job to prove that is acting on Alice's behalf. Normally, Alice would directly use her long-term X.509 End Entity Certificate[1] in order to delegate the proxy certificate[2] to the job. However, as she is not in her office today (and therefore she does not have access to her End Entity certificate), she instructs the University Grid Portal to retrieve such a proxy certificate from The University MyProxy [12] online credential repository and to use it to delegate a proxy certificate to the job running on the University HPC Center Linux Cluster. Once the job is executed, it will set up the Navy Institute Wave Tank, retrieve and correct the simulation data and save it to the University Mass Storage Solution.

For each remote resource involved, the job has to authenticate (achieving single sign-on through its delegated proxy certificate) and be authorized (the identity of the certificate is checked against access control lists, for example by mapping the identity of the certificate to a local account by means of a gridmap file). In addition, Alice has been informed that the Navy Institute requires a proof of her University affiliation before being granted access to any instrument. Therefore, Alice has programmed the job to contact the University Community Authorization Service(CAS) [13] and retrieve, after another round of authentication and identity based authorization, a statement attesting Alice's involvement with the University. Using this certificate and assuming that all the other Wave Tank local requirements are fulfilled (the Wave Tank is not in use and the University has not exceeded its allocated hours) authorization is granted at the Navy Institute site. Finally after one more authentication and authorization round with the University Reliable File Transfer Service the corrected and refined data is stored, being available to Alice and her group.

Although it seems that this scenario (depicted in figure 1) fits Alice's and the resource owners' needs, there are some implicit assumptions which are necessary for Alice's job to succeed:

[1] An End Entity certificate is a long-term certificate issued by a certification authority and therefore, its private key must be stored securely to prevent unauthorized access.

[2] A Proxy certificate is a certificate issued by an End Entity certificate or another proxy certificate with a lifetime of several hours. Due to this short lifetime, it is considered safe to store its private key unencrypted, protected only by file system local permissions.

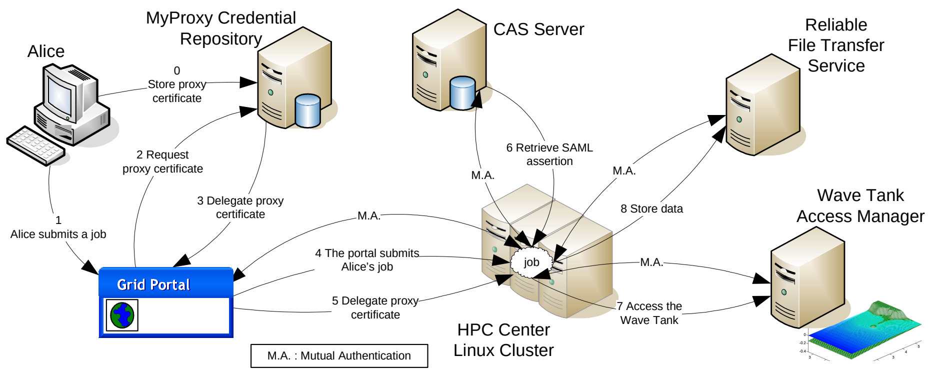
MyProxy Credential Repository
Alice
0 Store proxy certificate
CAS Server
Reliable File Transfer Service
2 Request proxy certificate
6 Retrieve SAML assertion
M.A.
M.A.
3 Delegate proxy certificate
8 Store data
1 Alice submits a job
M.A.
Wave Tank Access Manager
4 The portal submits Alice's job
job
M.A.
Grid Portal
5 Delegate proxy certificate
7 Access the Wave Tank
HPC Center Linux Cluster
M.A. : Mutual Authentication

Fig. 1. Grid Scenario

– One credential for all. A job is submitted together with a proxy certificate signed by a Certification Authority (CA) which is further used to authenticate to other resources. This assumes that all resources trust the same CA.
– Identity Based Authorization. Resources, where a job is allowed access, have to know in advance the identity of the certificate.
– Simple Authentication/Authorization. It is based on a one-shot process where the job requests access and the resource grants or denies it.
– Manual Credential Fetching. Users need to find out in advance which credentials are required to access each resource and program their jobs to fetch and give these credentials while authenticating/requiring authorization.

These requirements become liabilities when the Grid grows more complex and the number of resources a job has to access increases. Mapping identities raises serious scalability problems due to the large number of potential users, even more when we take into account the difficult requirement of having a single trusted Certification Authority (which is hard enough to have within one Grid and not feasible when trying to integrate different Grids). Because of these reasons, property based certificates have started to appear (PRIMA [10], VOMS [1], CAS [13] and X.509 attribute certificates [5]) even though there is no standard interface for using them yet. Furthermore, access control is not a simple task, as both a job and a resource might require to specify constraints in the way they disclose their certificates. This asks for extension of the usual one-shot mechanism to an iterative process where the level of trust increases at each iteration [22]. Finally, resources should advertise the credentials they require thus allowing the job to perform dynamic credential fetching. This not only frees job owners of "what resource requires what credential" problem [3] or that of coding credential fetching within their jobs but also allows dynamic selection of resources as they would be self explanatory in terms of authorization requirements.

[3] Currently, users must keep track of the right/required credential for each resource they might access. Application needs may be difficult to predict prior to its execution and the user might not be available when a certain credential is required. On the other hand providing the application with all user credentials is not feasible as it may imply revealing sensitive information.

We argue in the following sections that Grid with support for specifying, advertising and enforcing service level access control policies together with capabilities for automatic fetching of credentials can indeed suit large scale collection of resources, enabling dynamic negotiation for authorization and access granting based on parties properties, and can be implemented on top of the Globus Toolkit 4.0.

3 Trust Negotiation

Distributed environments like the Web, P2P systems or Grids are built with the assumption that service providers and consumers are known to each other. In common scenarios before allowing access to (possibly) sensitive resources, entities establish trust relations by having clients pass a registration phase. This registration phase consists of the creation of an account, the addition of the user identity to an access control list or some offline contact specifying who provides the service, under what conditions and to which consumers. This becomes a liability due to the lengthy process the registration phase supposes (resource administrators often have to manually verify user data e.g. associations with institutions or projects), which delays resource usage and restricts its availability even if the client is entitled to access it. Moreover clients are required to reveal sensitive information (name, residence, position or credit card number) with no mean of imposing their own requirements checking the service provider reliability.

Trust relations are constructed based on a set of digital credentials (e.g. certificates, signed assertions, capabilities or roles) which are disclosed by two entities to prove certain properties they pose. Digital credentials bind the identity of the holder to a public key, are signed by a credential issuer and may attest a quality of the holder. Since the credential may include personal information (e.g., roles and capabilities) users should be entitled to have their own requirements with regard to the entity to which this information is released. In such a case, a more adequate approach would be that distributed environments treat clients and service providers equally, similarly to P2P systems, both having the ability of imposing restrictions on resource disclosure (being resource services or credentials).

Trust negotiation [22] provides a gradual establishment of trust between parties through requests for and disclosure of credentials in an iterative and bilateral process. The requests for credentials can be viewed as the resource authorization policies, enabling upon satisfaction the authorization decision for accessing the resource. Trust negotiation approach distinguishes from the identity based access control in the following regards:

- Trust is established between previously unknown parties based on their properties.
- Providers and consumers can define policies to protect their resources (credentials or services provided).
- Authorization is established incrementally through a sequence of mutual requirements and disclosures of credentials.
- The authorization process may involve more than two entities and their trusted credential issuer, as requirements for a credential may determine a new negotiation with a third entity, which at its turn may call for another negotiation and so on.

Trust negotiation is triggered when one party requests to access a resource owned by another party. The goal of a trust negotiation is to find a sequence of credentials

$(C_1, \cdots, C_k, R)$ where R is the resource where access is attempted, such that when credential C_i is disclosed, its access control policy has been satisfied by credentials disclosed earlier in the sequence, or to determine that no such credential disclosure sequence exists.

Having introduced the concepts behind trust negotiation we concentrate in the next sections on our vision of a policy based trust negotiation mechanism which enables an automated authorization scheme over a Grid environment.

4 Dynamic Negotiation for Grid Services Authorization

Our initial example (illustrated in section 2) involved two distinct domains sharing resources over a Grid: one administrated by the University and the other by the Navy Institute. University resources (Grid Portal, MyProxy Credential Repository, Linux Cluster, CAS) have been carefully setup to interoperate and previous contacts with the Navy Institute have established the requirements for addressing its instruments. In this section we provide a more flexible scenario able to accommodate a large, loosely coupled Grid environment, reducing the current management overhead.

We propose a scheme in which entities advertise their authorization requirements as access control policies and are able to query for these policies and fulfill them through automatic credential fetching. Credentials can be protected by policies that have to be satisfied by the requesting part before having them revealed. Acting in a self describing environment, services and clients will automatically negotiate authorization by iteratively increasing their trust relationship, through credential disclosures until a decision regarding authorization can be made. The following scenario reflects these new functionalities:

Alice attends a Grid Conference in another country. She can not log into the University Grid Portal as for security issues it can only be addressed from the University network. Luckily the conference organizers provide a policy enabled Grid Portal and Alice can use it to submit her job. She logs into the Conference Grid Portal with her registration number and instructs the portal to obtain a delegated certificate from the University MyProxy Server.

Policies set up by the Conference Grid Portal administrators permit user requests for delegated certificates if the user can prove to be registered at the conference and the MyProxy Server contacted belongs to an accredited university. The University MyProxy Server has no problem in disclosing a credential signed by the State Ministry of Education attesting University accreditation, but when asked to delegate a credential on behalf of a user, the University MyProxy Server has its own requirement as to receive a credential signed by IEEE or Verisign. The Conference Grid Portal has an IEEE signed credential attesting its organization under IEEE supervision but first has to check whether this certificate is protected by a policy. Indeed this is the case. The policy states that the entity asking for the IEEE credential has to prove her affiliation with a university. This has already been achieved through the certificate previously disclosed by the University MyProxy Server so the IEEE signed certificate is disclosed and negotiation succeeds with the Conference Grid Portal retrieving a credential for Alice's job.

For performance issues, Alice decides to submit her job to a Linux Cluster belonging to the Research Center for Aeronautical Sciences. For the job to be submitted, a

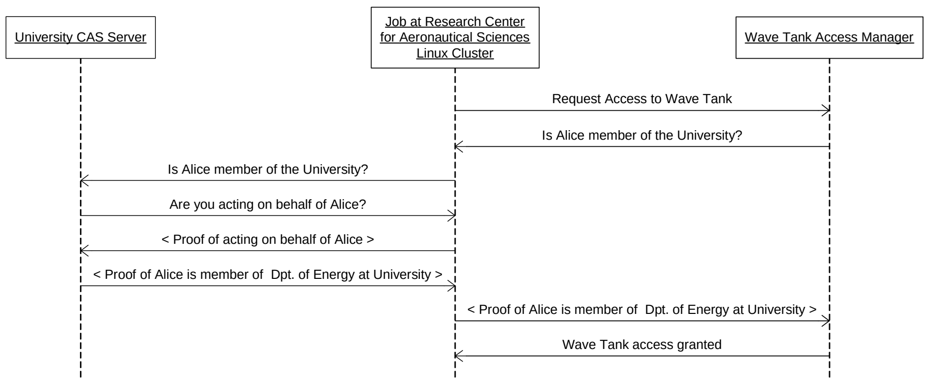
University CAS Server
Job at Research Center for Aeronautical Sciences Linux Cluster
Wave Tank Access Manager
Request Access to Wave Tank
Is Alice member of the University?
Is Alice member of the University?
Are you acting on behalf of Alice?
< Proof of acting on behalf of Alice >
< Proof of Alice is member of Dpt. of Energy at University >
< Proof of Alice is member of Dpt. of Energy at University >
Wave Tank access granted

Fig. 2. Simplified sequence diagram of the job negotiations

new round of negotiation takes place as the Linux Cluster policies require the client to provide a credential attesting her acting as member of a project present in the State Ministry of Education database. The Conference Grid Portal forwards a query to the State Ministry of Education CAS and, by providing Alice's job credential, retrieves a certificate attesting that Alice participates in a project regarding signaling instrumentation. By using this certificate, job authorization is granted at the Research Center for Aeronautical Sciences Linux Cluster. As the job needs to contact further resources, a Proxy Certificate is delegated by the Grid Portal to the Linux Cluster.

The job resumes its work by querying Navy Institute policies for Wave Tank access (job negotiations are represented in figure 2). It finds out that it has to prove Alice's association with a university organization. To demonstrate this, the job contacts the University CAS server and retrieves an assertion attesting that Alice is member of Department of Engineering at her university, not before proving to the CAS server that it is acting on Alice's behalf. Again we assume that the other local Navy Institute policies are fulfilled (the number of hours allocated to the University has not been exceeded and the Wave Tank is not in use) so the job can setup the Wave Tank and start receiving data.

For storing the output data with the University Reliable File Transfer (RFT) Service, the job has to provide a credential signed by the University Certification Authority (CA). Such a credential has been previously delegated to the Linux Cluster and, by providing its chain of certificates [4], the job meets the University RFT Service requirements and it is allowed to store the results without further negotiation.

The job has been submitted with only one credential, but then dynamically negotiated authorization at each resource accessed and whenever required, knowing where to retrieve the needed credentials. This was possible due to the contacted grid services ability to advertise policies, to indicate what credentials are needed and to specify where to retrieve them from. Resources were shared with no implied previous interactions with

[4] The certificate chain has at its root Alice X509 End Entity Certificate issued by the University CA

entities interested in accessing them, and administrators involvement reduces to setting policies for access control and credential disclosure.

5 Architecture Overview and Implementation

This section presents an extension to the recently released Globus Toolkit version 4.0 (GT 4.0), which allows advanced access control mechanisms and policy based negotiations, as depicted in figure 2. One of our main design goals was easy integration with current grid services paradigms and, therefore, our extension is backwards compatible and provides a straightforward installation with almost no additional effort.

The GT 4.0 Authorization Framework [9] supports grid service pluggable Policy Decision Points (PDPs) [9]. A PDP intercepts client calls and is responsible for authorizing them. It must return an authorization decision which can be either "granted" or "denied". In addition, a chain of PDPs can be specified for authorization allowing a series of verifications to be performed. In this case, the final decision is the conjunction of all of them, that is, all of them must return "granted" in order to permit access. This introduces a set of limitations as configuring custom PDPs for each access policy forces the client to follow only one chain of requirements and be granted access only if all policies have been satisfied. We desire to support also disjunction in the authorization decision and in this way provide to a larger number of clients the possibility of accessing the resource through the satisfaction of one chain of policies from several (presumably) available with the resource. This boosts the authorization mechanism by allowing resources administrators to have a single point of configuration for all authorization policies and permitting access to clients holding different sets of attributes. Moreover, because clients are offered several options for authorization, they may be able to choose according to their preferences (expressed in their own policies) which credentials are to be disclosed (e.g., credentials available locally or revealing less sensitive information).

Due to the restricting support of GT 4.0 in terms of service PDPs decision aggregation, we have developed a custom Interceptor PDP responsible for client call filtering and checking of a successfully completed negotiation process while the policy requirements and proofs are handled outside the PDP through the integration of a Negotiation Module into the service functionality. Check figure 3 for an architectural overview.

The Interceptor PDP allows client calls to a service in case no extra requirement is needed for the requested operations. Otherwise, they are denied with a negotiation exception sent to the client, until a successful policy based trust negotiation is completed. The Interceptor PDP and the operations which can directly be accessed without negotiations are configured through a security descriptor pointed out in the service deployment descriptor.

If authorization is denied, the client can start a negotiation with the service, having his negotiation module contact the service negotiation module and requesting access. The service sends back the access policy for the operation requested and the client can answer disclosing the credentials specified in the service policy or sending its own access policy for those credentials.

A grid service describes the operations supported and their parameters through its WSDL file. At the WSDL level the service is identified as a port type having a name

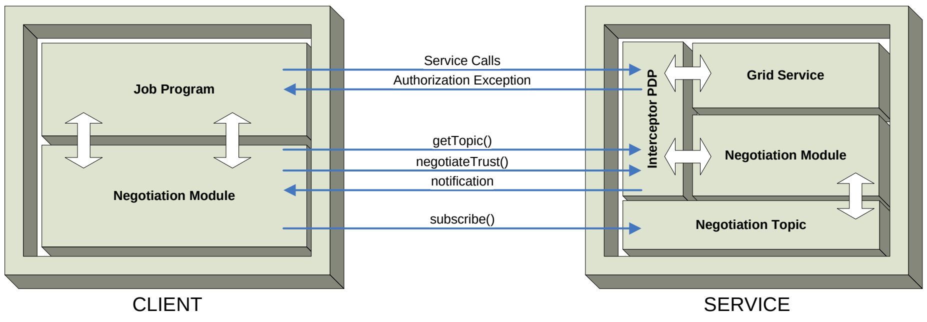

Fig. 3. Architecture Overview

and several operations, and using messages to carry input and output data. GT4.0 provides a useful feature in writing WSDL code for a grid service, the WSDLPreprocessor namespace which contains a tag "extends" permitting the inclusion of definitions of other port types in the current grid service definition. Using this feature it is possible to include the functionality of a service (giving that this functionality is also implemented in code) into another service. We have defined a WSDL file for a port type called Trust-Negotiation, which describes messages targeted for trust negotiation and exposes two extra operations (getTopic and negotiateTrust) enabling the exchange of policies and proofs between clients and services. By extending the TrustNegotiation port and having the functionality of the two extra operations implemented in the Negotiation Module, a general grid service is enhanced with trust negotiation capabilities. We emphasize that in order to confer this functionality to a usual grid service we have to modify only its descriptors (that is, configuration) files.

The disclosure of a policy from one entity to another may result in a credential being fetched from a service which at its turn may result in another policy based trust negotiation process, making it difficult to predict when one request can finish its evaluation, and thus to estimate how long a client has to wait for the negotiation process to conclude. In order to overcome this problem we used the GT 4.0 support for WS-Notification [23] as a mechanism for asynchronous client information of the grid service decisions: policies required to be satisfied, proofs disclosed or authorization reached.

The WS-Notification family of specifications includes WS-BaseNotification [18] and WS-Topics [20], standardizing asynchronous communications between consumers and providers. WS-Topics specifies how topics can be organized and categorized as items of interest for subscription. We associate each trust negotiation (triggered by the client request for a service operation) with a topic of interest through which messages can be delivered to the client side. Consumers register themselves with a certain topic of interest, while providers deliver the notifications to the consumers, each time the topic has changed. Notification producers (in our case grid services) expose a subscribe operation through which, each consumer (either grid service or client) can register with a certain topic. Notification consumers expose a notify function that producers use to send notifications. The grid service use of notifications is configured also, through the service descriptors and their functionality is supported by the GT4.0 container. The functionality behind topic assignment, management and client registration for service notifications is implemented in our Negotiation Topic Module (figure 3).

In our design, each client interested in negotiating trust calls the getTopic operation (exposed by the grid service through the extension of the TrustNegotiation port)in order to receive a unique namespace associated with a topic [20] and subscribes to it. Service side policies and disclosed credentials reach the client asynchronously as notifications, while client policies and proofs are pushed to the service through client invocation of the trustNegotiate operation (also exposed by extending the TrustNegotiation port).

The use of the GT 4.0 implementation of the notification paradigm imposes one limitation to the grid service, that of exposing its state as a "Resource" in conformity with the Web Service Resource Framework (WSRF) [19]. "Resources" are used for storing a web service state from one invocation to another. Thus the only requirement we impose for integrating our policy based architecture for grid service authorization is having the service use such a "Resource". The addition of a resource to a grid service is straightforward, with only minimal additions to its descriptor files and code (only a matter of having the service implement the interface *Resource* with no methods).

The client has a complementary functionality to that of a grid service. For client programming we have developed a jar file and an API, to facilitate easy integration of trust negotiation capabilities to client code. The client has to use one class (GridClientTrust-Negotiation) for setting the grid service address and the namespace of the negotiation topic received. Also the client has to hide the service invocations by implementing one interface (SendWrapper), part of our API, for using the stub of the service with whom he wants to negotiate trust.

Service stubs mask the intricacies of communication with a service and can be generated automatically from the grid service WSDL file. In Java, stubs are represented as objects with methods for each operation exposed by the service. Since each stub is specifically generated for the service it targets to invoke, we have developed an interface (SendWrapper) abstracting the calls made to a trust negotiation enabled service stub (supporting our defined getTopic and negotiateTrust operations). The client has to implement this interface and one of its functions (sendMessage) to call the trustNegotiate operation on the stub. By implementing the same interface (SendWrapper), we have provided to the client an object/class he can use for automatic fetching of credentials from a CAS server.

On the client side a thread registers with the topic returned by the grid service and listens for notifications. SendWrappers are used on the client side for sending queries to the service with which the negotiation process is undergoing, or to third party services (e.g., CAS) for credential retrieval. The same client code can be used by the service to register for negotiation with other services or to retrieve credentials, as one service may be the client of another service with whom it negotiates trust.

Credentials used in our implementation may virtually be expressed under any kind of format (certificates, signed assertions, signed RDF statements) as the policies and the negotiation modules are decoupled of credential types. Currently we have support for X.509 v3 Certificates holding in extensions the owner attributes. In addition to this, we have integrated the use of proxy certificates which carry SAML Assertions [15] retrieved from a Community Authorization Service (CAS). A SAML Assertion indicates its issuer, the subject of the statement, a resource, and an action allowed on this resource. The client can wrap such a SAML Assertion in a proxy certificate and push it as a proof satisfying a policy disclosed during a trust negotiation process.

The architecture we propose is general enough to deal with different policy languages and policy evaluation procedures, which are implemented in the negotiation module of each entity. In our implementation we use a negotiation module based on the PEERTRUST project and language [6] with built-in support and verification of X.509 v3 certificates. The entire implemention is done in Java and uses an inference engine to reason over policies and credentials. We have tested our extension accessing different Grid services and automatically fetching credentials from a CAS server with positive results. Our implementation is distributed as a jar file which together with the small changes to the appropriate configuration files allows users and administrators to easily integrate it with current GT 4.0 grid services.

6 Related Work

In this section we provide a brief overview of related approaches to Grid authorization. The common characteristic of these authorization schemes is the replacement of identity based access with the usage of user attributes in the authorization process. In this regard, the major improvement comes from the increased number of entities able to be accommodated by a Grid environment released of the consistent mapping of client identities to user accounts. Nevertheless, despite its benefits for increased scalability, this approach still places on the client side the burden of appropriate certificate provision and retrieval for every accessed location. In addition, the authorization schemes presented further, tend to be client centered, with attribute requirements and their demonstrated possesion enforced only for the client side and no policy imposed to the service itself.

Virtual Organization Management Service (VOMS) [1] supports attribute based access control to the Globus Toolkit Resource Allocation Manager (GRAM) responsible for job execution. Clients retrieve pseudo certificates containing client attributes (groups and roles) from VOMS compliant servers and include them in a non-critical extension of a usual proxy certificate. Such proxy certificates are pushed to the resource together with the submitted jobs, and based on their contained attributes an authorization decision is made.

PERMIS [3] project has as its main goal the construction of an X.509 role based Privilege Management Infrastructure that could accommodate diverse role oriented scenarios. PERMIS consists of two subsystems: the privilege allocation subsystem which issues user X.509 Attribute Certificates and stores them in LDAP directories for later retrieval and the privilege verification subsystem which pulls the user certificates and the policies regarding user roles from a pre-configured list of LDAP directories (clients can also push certificates to the privilege verification subsystem).

AKENTI [16] uses distributed policy certificates signed by stakeholders from different domains in order to make a decision regarding access to a resource. Akenti uses an XML format for representing three types of certificates: *attribute certificates* binding an attribute-value pair to the principal of the certificate, *use-condition certificates* indicating lists of authoritative principals for user attributes and containing relational expressions of required user attributes for access rights and *policy certificates* consisting of trusted CAs and stakeholders issuing use-condition certificates and lists of URLs from where use-condition and attribute certificates can be retrieved. Akenti engine authenticates clients based on their X.509 certificate, gathers in a pull model certificates available with the authenticated identity (attribute certificates) and those of the accessed resource (use-condition certificates) and makes a decision regarding user rights at the resource.

System for Privilege Management and Authorization (PRIMA) [10] manages and enforces privilege and policy statements expressed in X.509 Attribute Certificates. In PRIMA design, resource administrator and stakeholder issued certificates are revealed by the client to the resource Policy Enforcement Point (PEP). The PEP validates user attributes and verifies with the resource Policy Decision Point (PDP) if the issuers are authoritative for user's presented privileges. All acknowledged privileges are gathered by the PEP in a policy further presented to the PDP for verification against the configured access control policies. The PDP returns to the PEP an authorization decision and a set of recommendations (file access permissions, user quotas and network access [11]) for setting up a local account with support for the user validated privileges.

Similar to the above discussed authorization approaches, our architecture accommodates user attributes for access granting, but in addition offers solutions for client query of resource access policies and their compliance through negotiations and automatic credential fetching. Furthermore we allow clients to define and enforce their own policies for service authorization.

Another area of research is concerned with the standardization of policy representation. Currently there is an extensive effort in enabling eXtensible Access Control Markup Language (XACML) [4] usage for authorization management in Grid environments. XACML is an OASIS standard for specifying access control policies and the associated request/response formats. It allows use and definition of combining algorithms which provide a composite decision over policies governing the access requirements of a resource. Future versions of Globus Toolkit are expected to be delivered with an integrated XACML authorization engine. However, current versions of XACML are not yet expressive enough to deal with some of the requirements for the integration of our approach (e.g. delegation of authority).

7 Conclusions and Further Work

This paper described current authorization mechanisms in Grids together with their limitations. We described how support for policy-based negotiation can be integrated into a Grid infrastructure, allowing advanced access control and automatic credential fetching, in order to provide a solution to most of those limitations. We discussed how to easily integrate these extensions into the Globus Toolkit 4.0 and presented our current implementation.

Our future research will focus on online credential repositories in order to extend the automatic credential fetching capabilities. We have already integrated CAS servers and we plan to do the same with MyProxy repositories. Unfortunately, these two types of online credential repositories do not share a common interface (requiring an ad-hoc integration by means of wrappers) and do not yet allow us to store arbitrary credentials which would be desirable for more complex scenarios.

We also investigate policy representation in XACML and the possibility of relying on the SAML support for querying and exchanging policies [14].

In addition, we are planning to perform more experiments in order to study the performance loss induced by policy integration into current grid services. However, although we know in advance that access will be somewhat more expensive than based on current approaches, we are certain that it will pay off as many of the tasks that have to be done manually so far can then be done automatically. On the other hand if we consider that Grid is designed to enable resource sharing for large computational jobs, whose requirements can not be met locally, an increased setup time is less significant compared to the ability of dynamic satisfaction of resource authorization policies.

Acknowledgments

This research was partially supported by the Network of Excellence REWERSE (`http://rewerse.net`, IST-506779)

References

1. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lőrentey, and F. Spataro. Voms: An authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference*, Santiago de Compostela, Feb. 2003.
2. J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett. Negotiating trust on the grid. In *2nd WWW Workshop on Semantics in P2P and Grid Computing*, New York, USA, may 2004.
3. D. Chadwick and O.Otenko. The permis x.509 role based privilege management infrastructure. In *7th ACM Symposium on Access Control Models and Technologies*, 2002.
4. eXtensible Access Control Markup Language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
5. S. Farrel and R. Housley. An internet attribute certificate profile for authorization, rfc3281.
6. R. Gavriloaie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic Web Symposium (ESWS 2004)*, volume 3053 of *Lecture Notes in Computer Science*, pages 342–356, Heraklion, Crete, Greece, may 2004. Springer.
7. Globus toolkit 4.0. http://www.globus.org/toolkit/docs/4.0/.
8. Grid security infrastructure. http://www.globus.org/security/overview.html.
9. Gt 4.0 authorization framework. http://www.globus.org/toolkit/docs/4.0/security/authzframe/.
10. M. Lorch, D. Adams, D. Kafura, M. Koneni, A. Rathi, and S. Shah. The prima system for privilege management, authorization and enforcement in grid environments. In *Proceedings of the 4th Int. Workshop on Grid Computing - Grid 2003*, Phoenix, AZ, USA, Nov. 2003.
11. M. Lorch and D. G. Kafura. The prima grid authorization system. *J. Grid Comput.*, 2(3):279–298, 2004.
12. J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: MyProxy. In *Symposium on High Performance Distributed Computing*, San Francisco, Aug. 2001.
13. L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
14. Saml 2.0 profile of xacml v2.0. http://docs.oasis-open.org/xacml/2_0/access_control-xacml-2.0-saml-profile-spec-os.pdf.

15. Security assertion markup language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.
16. M. R. Thompson, A. Essiari, and S. Mudumbai. Certificate-based authorization policy in a pki environment. *ACM Trans. Inf. Syst. Secur.*, 6(4):566–588, 2003.
17. S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. *Internet X.509 Public Key Infrastructure Proxy Certificate Profile*. http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-10.txt, Dec. 2003.
18. Web services base notification. http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf.
19. Web services resource framework. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.
20. Web services topics. http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.pdf.
21. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. In *3rd Annual PKI R&D Workshop*, Apr. 2004.
22. W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. DARPA Information Survivability Conference and Exposition, IEEE Press, Jan 2000.
23. Ws-notification.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn.