Towards a Common Authorization Infrastructure for the Grid

Krzysztof Benedyczak[1,2], Marcin Lewandowski[1], and Piotr Bała[1,2]

[1] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Chopina 12/18, 87-100 Toruń, Poland
[2] Interdisciplinary Center for Mathematical
and Computational Modeling
Warsaw University
Pawińskiego 5a, 02-106 Warsaw, Poland

Abstract. In this paper we report results of the ongoing effort to provide a seamless authorization for the UNICORE and Globus Toolkit middlewares using the UNICORE Virtual Organizations System (UVOS). The UVOS is already well integrated with the UNICORE middleware. We have designed and created a set of native Globus Toolkit 4 modules which enable a UVOS based authorization, with a similar functionality as its UNICORE counterpart. Actually the same authorization data stored on the UVOS server can serve both middlewares simultaneously. The paper provides an overview of existing approaches to the user management problem in the Grid environment with a special emphasis on those which can be used across different grid middlewares. The paper presents the UVOS system, its features and how its adoption helps to manage users of the UNICORE and Globus 4 middlewares.

1 Introduction

The problem of user management in the Grid is deliberated from the very beginnings of the Grid concept. The fundamental ideas come from the publication [1], which models the society of grid users as *virtual organizations* (VO). Over time numerous implementations were developed including a popular Virtual Organizations Management System (VOMS) [2]. In addition to the centralized VOMS, an idea of federations started to gain interest, mostly because of the success of the Shibboleth system [3] used to authenticate and authorize web users. Nevertheless the engineered solutions still contain a number of shortcomings. Primary problem is a complicated administration, lack of a fine-grained authorization and a complex deployment procedure. As there are numerous user management systems available, one of the most important problems now is the interoperability, especially when a homogenous authorization shall be provided across heterogeneous grid middlewares.

The goal of the work reported in this paper is to create a native support in Globus Toolkit 4 for user authorization, based on their data stored in a UVOS

H.X. Lin et al. (Eds): Euro-Par 2009 Workshops, LNCS 6043, pp. 325–334, 2010.

server. UVOS server is a part of the UNICORE Virtual Organizations System [4] and it is a modern VO solution developed mostly for the UNICORE grid system. However UVOS is not tightly bound to the UNICORE as it employs service oriented architecture paradigm. It uses open Security Assertion Markup Language (SAML) 2.0 protocol for communication [5] which allows for easy integration with different grid middlewares.

The native support for the UVOS system in the two significant grid middlewares can be seen as a big step towards a full grid interoperability in the area of users authorization. It is even more important as the SAML protocol used by the UVOS is utilized in the aforementioned Shibboleth federation management system. Therefore it is logical to expect that administrators will be able to switch effortlessly between those different authorization solutions or even use them together.

The next part of the work discusses the common methods of integration of the virtual organizations with the grid systems mostly using the UNICORE and Globus Toolkit middlewares as examples. The subsequent section presents in more details the UVOS system. The main part of the work — prototype of the UVOS authorization modules for the Globus Toolkit — is described in the section 5. The paper is concluded with a summary of the achieved results and an enumeration of ideas for a future work.

2 The Grid Approach to the User Management Problem

The basic mechanism of user management in most of the grid middlewares is usage of a database placed locally to a grid site (plain text file in the Globus Toolkit, a simple SQL database in UNICORE). More advanced solutions are using one of the two common approaches: the grid-central databases of users or federations. Both allow for creation of Virtual Organizations i.e. the situation when users coming from different real organizations get access to the shared resources, often located in different administrative domains.

The global database approach is simpler in installation and deployment. However it requires that central database must allow for the administration of its content (or more strictly speaking its fragments) by multiple managers with different permissions. There is also a problem of reusing users data stored in already existing catalogs such as Lightweight Directory Access Protocol (LDAP) servers. Those problems are addressed by the federation concept, where users are always authorized by their home (or real) institution. This approach is much more scalable but also more difficult to set up: it requires a mutual trust (often formal) between all user privileges providers and resource providers. A common format and semantics of users privileges must be developed and applied.

There are two practical models of establishing users authorization data used in the grid systems: *a pull* and *a push* models which are presented in the figure 1. One should note that both models may be applied simultaneously.

In the *pull mode* a service (e.g. a grid node server) contacts the VO server to obtain the attributes of a user who tries to use it. The attributes received from

the VO server can be used for an authorization, e.g. server's policy may permit only those users who possess certain attributes. The service may use received attributes to perform other tasks e.g. to map requester to a local UNIX account. Pull mode is transparent for the grid users. However it is more difficult for grid administrators to set it up: every grid site must be correctly configured to use the VO server.

In the *push mode* a user has to contact a VO server on her own and get the list of possessed attributes in a signed assertion. Later this assertion can be attached to the requests which are sent to the grid services. If the service trusts the assertion issuer then it can use the attributes for authorization. The user can usually ask the VO server for a subset of owned attributes. In such a case the user can hide a part of her identity or alter the execution (e.g. by choosing role). The push mode is more scalable in terms of server administration and easier to set up. However it requires user interaction and thus is more suitable for advanced grid users. Also for the push mode a problem with expired assertions arises.

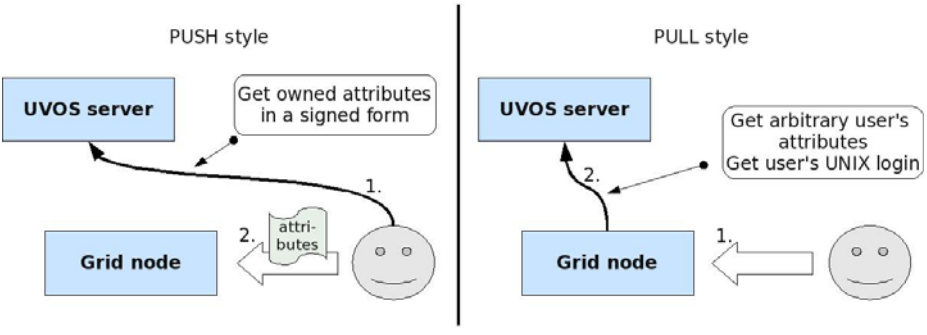

Fig. 1. The figure shows the two alternatives for provisioning client's attributes to the grid node: by the service (pull mode) and by the client (push mode)

3 Related Solutions and Efforts

One of the most important developments in the area of authorization is the GridShib project [6] based on a standard protocols. It was created for the Globus Toolkit. The primary goal of the GridShib project is creation of a full-fledged tool which integrates the Shibboleth Identity Provider as the information source about the users for the Globus middleware.

The system developed by the GridShib project is a production ready solution with a vast amount of features. It allows for using both push and pull styles of attribute retrieval. In the first case it can consume attribute assertions attached to a client's certificate (both normal X.509 and proxy). Additionally, recent releases of the GridShib software support VOMS assertions. What is interesting, the GridShib provides a modified version of a mechanism which establishes local user names for the grid clients. Instead of direct mapping of client's certificate subject name to a local account, GridShib allows for choosing the local name

based on the client's attributes. The attribute to handle local name mappings are stored in a configuration file, local to the grid node.

Unfortunately GridShib can cooperate only with the old, 1.x releases of the Shibboleth system. It cannot be used with the current SAML 2.0 protocol and therefore it can not be used with the Shibboleth 2.0 or UVOS systems. Anyway we can foresee that GridShib would become the fundamental solution used for interoperable authorization of the Globus Toolkit, when it will be upgraded to the SAML 2.0.

GridShib possesses also an another shortcoming very important from the practical point of view. Namely, it is impossible to authorize clients of the legacy Globus services. Legacy services are not yet converted to the web services technology and are not OGSA [7] based. In particular there is one such a service of a critical importance: the GridFTP file transfer server. As a result GridFTP access must be based on the traditional gridmap file and there is no possibility for a complete user management via the Shibboleth federations.

In result, authorization stack for the Globus Toolkit must be implemented in two flavors: for the pre-WS and WS components. A common approach to overcome this problem is to build or even download a full gridmap file from the remote server. One of the popular tools is `edg-mkgridmap` from the Virtual Data Toolkit project [8]. Another example is GUMS software [9] which is much more advanced. Those tools, quite interesting for the Globus Toolkit are not a good base for the interoperable solution because both are using simple "user import" approach which has general problems such as low performance with large amount of users.

Along with the development of different user management systems for the grid, various efforts were undertaken to integrate them. One of the largest is the IVOM (Interoperability and Integration of VO-Management) project, a part of the D-Grid initiative. The IVOM aims to develop services that enable integration of VOMS and Shibboleth-based VO management systems with the grid middlewares used in Germany, in particular gLite, Globus Toolkit 4 and UNICORE 5. As it is explained in the publication [10] for this purposes only the push model is used. The paper suggests development of SAML enabled components for the grid middlewares which are in the project's scope. In the case of Globus Toolkit it is suggested to use the GridShib system. The paper [10] states that unfortunately GridShib does not support legacy Globus services.

4 The UVOS Overview

The UNICORE Virtual Organizations System (UVOS) is a new solution for centralized VO management. It was created in course of the EU-funded Chemomentum project [11]. The fundamental aim of the project was to develop a solution which will overcome the two important adoption blockers of a VO software: lack of flexibility and difficult deployment. The UVOS principles are:

– *Distributed environment*: The single installation can be completely controlled remotely.

- *Openness*: The system consumers can communicate with it using open and well established protocols.
- *Easy of Use*: The system can be easily installed and managed.
- *Flexibility*: The system provide tools and features which will make it useful in both OGSA (so SOA) environments and WWW environments.

UVOS architecture uses a central UVOS server which acts both as authentication service and attribute authority. The server is used by two kinds of clients: consumers and management clients. Consumers do not modify the UVOS content but query it. Management clients are used to dynamically modify VO data either by VO administrators or other management software .

The UVOS server (as the whole system) is written purely in Java so it is highly portable. All operations of the UVOS server are available via the web services interface. The server uses relational database to internally store the whole data, therefore it does not depend on any external services like LDAP.

The UVOS consumers use the open standard SAML 2.0 as a protocol to communicate with the UVOS server. The server implements the core SAML specification and to ensure a higher interoperability level it implements additional profiles:

- XACML Attribute Profile [12].
- SAML Attribute Query Deployment Profile for X.509 Subjects [13],
- SAML Attribute Self-Query Deployment Profile for X.509 Subjects [13],
- OGSA Attribute Exchange Profile Version 1.2 [14],

Finally, the UVOS web server is truly extensible by means of classic Java web applications (servlets), which can be simply installed by copying them into a designated server's installation directory. Two such web applications extending server's functionality are provided as ready to be used modules: one provides authentication form for the SAML authentication request protocol, the second one supports enrollment of new users.

UVOS access is restricted by it's own authorization stack. No external components/services are used to perform authorization. The authorization mechanism is advanced and provides a complete control of access on a group level.

4.1 UVOS VO Model

UVOS organizes VO members within a hierarchical group structure. Top level groups of this structure are called virtual organizations however are not different than other groups. Each entity can be a member of an arbitrary number of groups. It may has assigned a set of attributes. In addition, a single entity can possess multiple representations, for example in different formats. These equivalent incarnations of the same entity are called identities, and are usually invisible for an outside user.

Group membership is inherited in UVOS. The member of subgroup becomes automatically the member of the parent group. This is different than e.g. in VOMS, but has been requested by the users.

Every entity has a unique label and one or more tokens that represent it. Tokens must be in one of the supported formats, which currently are:

- a full X.509 certificate,
- an X.500 distinguished name,
- an e-mail address with an password used for authentication.

A token along with it's type is called an identity. As explained, an entity typically possesses one identity, but it can also has more. This reflects the real life situation where the single user can possess multiple certificates and email accounts. Additional identity formats may be added to the UVOS system with a intermediate level of effort. It is worth pointing out that all of the identities that compose an entity share the same characteristics (attributes, group membership, permissions, etc.): the UVOS works using entities internally.

UVOS attributes are composed of a name and a list of values. A name is a URI, and values are arbitrary strings. The value list can be empty. UVOS allows for three different ways of attributes assignment:

- global attributes: an entity can have an attribute assigned globally. Such an attribute is valid always and in every context,
- group-assigned attributes: an attribute can be assigned to a group, in which case all members of this group automatically hold this attribute (no matter if they were added later or prior to the creation of the group-assigned attribute). It is worth pointing out that this attribute is valid only in the scope of this group,
- group-scoped entity attributes: those attributes are assigned to the entity, just like global attributes, but have an additional group restriction and are valid only in in the scope of the group.

The last two methods introduce a "group-scoped validity" of attributes, which requires a further explanation. Within this mechanism the requester can ask (using the API provided by the UVOS service) for the entity's attribute either globally or valid only in a specified group. Global query returns global attributes only. A query limited to a group will return all entity's global attributes and all group-scoped attributes valid within the specified group.

4.2 The Client Side

In the UVOS currently there are available two management clients: the command line client (UVOS CLC) and VO Manager. The command line client can be used to administer UVOS from the console. It can be used in an interactive or batch mode. The UVOS VO Manager is a powerful GUI application based on the Eclipse Rich Client platform. It is much easier to use than a command line client, offers an intuitive interface so usually UVOS VO Manager a preferred choice. The VO Manager application can be considered as one of the most important advantages of the UVOS.

UVOS is available in the standard distribution of the UNICORE 6 middleware. Both push and pull modes are supported. In the push mode user can get

and attach certain attributes using the UNICORE Rich Client plugin. The pull mode is implemented as the module of the UNICORE server.

5 Globus Support for UVOS

Globus Toolkit 4 provides a rich interface for building and plugging additional authorization modules. Unfortunately, as it was briefly noted above, the Globus Toolkit is currently built using two different technologies. The main part of Globus employs a web services technology and is deployed in a special Java container. The rest of services, including the important file transfer subsystem GridFTP, is programmed in C and still does not use web services technology. Those services are called as *legacy* or *pre-WS*. Obviously the authorization APIs for both types of services are absolutely different. More advanced and full of features is interface provided by a Java web services container. To make a situation even more complicated, the web services version of authorization API was greatly refactored with the introduction of the version 4.1 of the Globus Toolkit.

The issues described above may be reason for a small popularity of the pull style solutions for the Globus Toolkit. The most of the popular approaches are focused on the push model. In the case of the Globus Toolkit pushed attribute assertions are embedded inside a user's proxy certificate. This pattern is used for both VOMS credentials and it is a typical usage scenario for GridShib deployments.

In our solution the pull model was chosen as a base for the implementation. The main reason for this decision was that client tools need not to be modified in any way for pull style. This is an important factor as Globus community is quite well established and in our opinion it would be hard to convince existing users to use an another client side tools or wrappers, as GridShib does.

The outcome of our project is a set of Globus Toolkit authorization modules. The modules allow for the pull style authorization based on the information stored in the UVOS server. The modules provide this functionality for *all* Globus Toolkit 4 modules, both 4.0.x series and more recent 4.1.x and 4.2.x. There are three principal functional components of our implementation:

- UVOS Policy Information Point which solely contacts the UVOS server and collects the information about the grid client.
- UVOS Gridmap Policy Decision Point allows administrator to store certificate to local account mappings in the UVOS server. This can effectively replace the traditional gridmap file. The mappings are stored as scoped attributes of a user, and a group (where the attribute is valid) is used to distinguish the mappings for different grid nodes if needed.
- UVOS Access Policy Decision Point which provides a possibility to perform a fine grained authorization decisions based on the attributes received from the UVOS server.

The Policy Information Point (PIP) and Policy Decision Point (PDP) terms come from the security domain and have a precise meaning in the Globus Toolkit

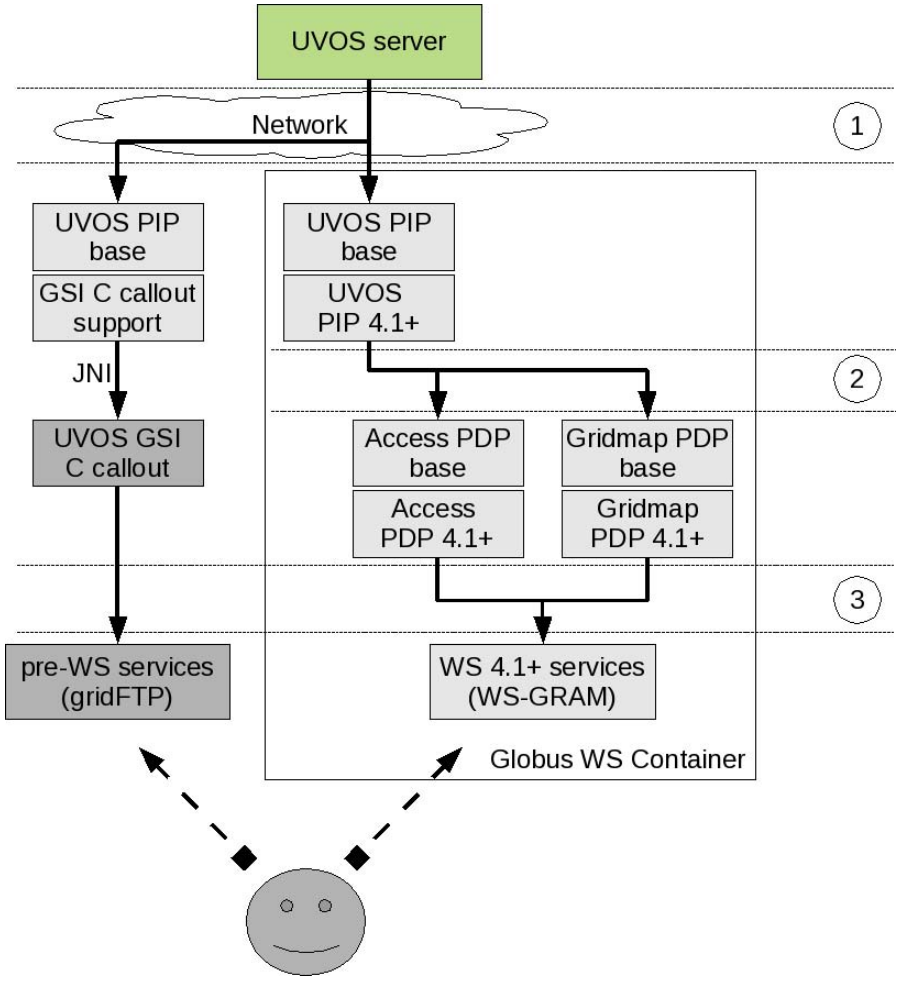

Fig. 2. The figure shows the architecture of the UVOS authorization subsystem for the version 4.1+ of the Globus Toolkit. All components which are specific to that particular Globus version are marked. Solid arrows shows information flow through the system. Dashed lines shows service invocations as performed by a user. Light boxes are Java components and dark gray boxes are C modules. In the figure there are three layers of communication marked. In the layer (1) communication is done through a network. The Policy Information Point (PIP) queries the UVOS server for the user's attributes. Communication in the next layer (2) is performed internally by the Globus security stack which passes assembled information about the client to the authorization modules. Eventually in the layer no (3) an authorization decision and (optionally) a mapping to a local account is passed again by the security stack to the invoked service (or more formally to the Globus policy enforcement code which protects the service access).

authorization API. The PIP is responsible for assembling grid client's credentials (as for example its identity, attributes) and the PDP makes an authorization decision (using the data provided by the PIPs). In fact this design is used only for Globus web services. The authorization API for the legacy services is much simpler — it allows only to create a callout which makes an authorization decision.

All three elements of our project are available for the Globus web services only. In the case of the legacy services only the UVOS Gridmap PDP element is available. It encapsulates the PIP functionality. The more fine grained access control would be clearly service specific so we do not plan to develop more features for our pre-WS Globus authorization module.

The security infrastructure of the Globus WS container allows for using multiple PIPs and PDPs together. The configuration is flexible with the changes introduced with the Globus Toolkit 4.1. Among others it is possible to choose an algorithm which makes a final access/deny decision taking as an input the decisions of the individual PDPs. As our modules may be installed individually we can achieve a high flexibility, e.g. by mixing UVOS and classic gridmap file authorization.

The implementation of our software consists of multiple small modules. The base implementation of the UVOS PIP was done using the Java language so it was possible to make use of the UVOS client library. As this implementation must work in different environments, there are three wrappers for the Globus Toolkit 4.0, 4.1+ and for the C security API. In the last case the Java Native Interface technology was used to connect both technologies. The PDP modules were designed in an analogous way, with the single exception of the code for the pre-WS services. In this case there is only one module which incorporates a C PIP wrapper. The whole architecture of our prototype is presented in the fig. 2.

6 Summary

We have designed and developed a prototype of a complete authorization solution for Globus Toolkit 4.x interoperable with the UNICORE 6. It uses the UVOS server as a backend. The pull style of attributes acquisition is used. One of the important results of our work is support for the GridFTP authorization which is absent in the GridShib system. Additionally, a native and deep integration with the Globus security stack allows for a more flexible usage of our modules with the other ones.

The future work will be carried out in multiple directions. In order to produce a production ready implementation we will have to provide solutions for reliability, that is solutions for the proper operation of the system during the failure of a communication with the UVOS server. Later plan to verify the interoperability of the system (and also the whole UVOS system) with the Shibboleth 2 middleware. Further integration of our solutions with the GridShib is planned but will be possible after a SAML 2.0 compatible version of GridShib will be released. Eventually providing a similar solutions for the other leading grid middlewares, gLite and NorduGrid ARC can be seen as an ultimate goal.

Acknowledgements. This work is supported by European Commission under IST grant Chemomentum (No. 033437). The financial support of the eea grant Kardionet (No. PL-0262) is also ackowledged.

References

1. Foster, I.T., Kesselman, C., Tuecke, S.: The Anatomy of the Grid — Enabling Scalable Virtual Organizations. In: Computing Research Repository (CoRR), vol. cs.AR/0103025 (2001), http://arxiv.org/abs/cs.AR/0103025 (2009)
2. Alfieri, R., et al.: From gridmapfile to voms: managing authorization in a grid environment. Future Generation Comp. Syst. 21(4) (2005)
3. The Shibboleth system webpage (May 2009), http://shibboleth.internet2.edu
4. The UVOS project (May 2009), http://uvos.chemomentum.org
5. Cantor, S., et al. (eds.): Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard (March 15, 2005), http://docs.oasis-open.org/security/saml/v2.0/ (May 2009)
6. Barton, T., et al.: Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy. In: 5th Annual PKI R&D Workshop (April 2006), http://grid.ncsa.uiuc.edu/papers/gridshib-pki06-final.pdf (May 2009)
7. Foster, I., et al.: The Open Grid Services Architecture, Version 1.5. GGF 2006 (2006), http://forge.gridforum.org/projects/ogsa-wg (May 2009)
8. The Virtual Data Toolkit project (May 2009), http://vdt.cs.wisc.edu/index.html
9. The Grid User Management System (May 2009), https://www.racf.bnl.gov/Facility/GUMS/1.3/index.html
10. Groeper, R., et al.: A concept for attribute-based authorization on D-Grid resources. Future Generation Comp. Syst. 24(3) (2009)
11. The Chemomentum project (May 2009), http://www.chemomentum.org
12. Hughes, J., et al. (eds.): Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard (March 15, 2005), http://docs.oasis-open.org/security/saml/v2.0/ (2009)
13. Scavo, T. (ed.): SAML V2.0 Deployment Profiles for X.509 Subjects, OASIS Committee Specification (March 27, 2008), http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml2-profiles-deploy-x509-cs-01.pdf (May 2009)
14. Venturi, V., Scavo, T., Chadwick, D.: OGSA Attribute Exchange Profile Version 1.2. Open Grid Forum (2007)