VIRTUAL ORGANIZATIONS AND DATABASE ACCESS – A CASE STUDY

Mikko Pitkanen[1], Marko Niinimaki[2], John White[2], and Tapio Niemi[3]
[1] *Helsinki University of Technology, Finland*
[2] *Helsinki Institute of Physics at CERN, Geneva, Switzerland*
[3] *University of Tampere, Finland*

Keywords: Grid computing, distributed databases, virtual organizations

Abstract: This paper presents a case study of using virtual organization technologies in database access. A virtual organization (VO) is a collection of people in the same administrative domain. A user can belong to many virtual organizations and have a different role (user, client, administrator,..) in each of them. An authorization of a user to different services within a VO is based on the user's identity and a service called a Virtual Organization Membership Service (VOMS) that maps these identities with roles.
The user's identity can be established in two ways. If the user communicates with the service using his web browser, the user's certificate must be included in the browser. Another possibility is to use a proxy certificate. There, in the proxy creation process, the program that writes the proxy adds the user's proxy certificate information about his participation in different VO's and his role in each of them.
In order to demonstrate using these VO proxy certificates, we have extended the functionality of Spitfire, a relational database front end. This involves assigning the user a database role (read/write/update) based on the VO information in his certificate. There is also a graphical user interface for creating the mappings between VO roles and database access roles.

1 INTRODUCTION

The recent interest and subsequent research in the Grid computing field has given rise to the concept of "virtual organizations". A virtual organization (VO) is a collection of people in some administrative domain. The user can be a part of any number of internal groups in their organization and have multiple roles in many organizations (Alfieri et al., 2003). A user is authorized to perform tasks on a computing Grid according to their VO affiliation and their role(s) within the VO.

The authorization process becomes more complex when the number of users and the number of possible roles of the users in the service increases. As services are usually distributed, a centralized service managing the authorization data is needed. This service is called a Virtual Organization Membership Service (VOMS) (Alfieri et al., 2003). The VOMS service is a front-end to a database where the information about the users is kept. The server maintains lists of groups, roles and capabilities that belong to each user. The VOMS is used to bind authorization information to a users identity (Alfieri et al., 2003).

When a VOMS service is used, the user invokes `voms-proxy-init` program (or equivalent) to contact the VOMS server. This program produces a proxy certificate containing the authorization information from VOMS service (Alfieri et al., 2003). When the user enters a service presenting his VOMS certificate, the service can find out the user's authorization information from the certificate. This removes the responsibility to maintain up-to-date information in every server.

The information about a user's VO and their role in it can be easily utilized in the context of distributed databases being accessed by a European DataGrid (EDG) product, Spitfire (Project Spitfire, 2001). In a typical case, if a Spitfire administrator knows a user's VO (e.g., an academic collaboration ACAVO) and their role within this ACAVO (say, an administrator), they can assume which databases the user should be able to access or modify. Using Spitfire, along with the edg-java-security package and a graphical user interface we can easily add a rule concerning the access rights of the user. This greatly simplifies the task of the database administrators, since they do not need to create a database access and grant different rights to each new person in the VO.

Foster, Kesselmann and Tuecke emphasize the role

Pitkanen M., Niinimaki M., White J. and Niemi T. (2004).
VIRTUAL ORGANIZATIONS AND DATABASE ACCESS – A CASE STUDY.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 514-517
DOI: 10.5220/0002602305140517

of virtual organizations in resource sharing in (Foster et al., 2001), and mention database access in one of their Grid scenarios. The OgsaDai consortium (OgsaDai, 2002) has defined an architecture for Grid database accesses and released an implementation of it. However, as far as we know, the European DataGrid project has been the first to design and implement a VO-based access management and implementation, that is described in this paper.

2 SPITFIRE AND EDG JAVA SECURITY

Spitfire (Project Spitfire, 2001) is a project of Work Package 2 within the European Data Grid Project. It offers a Java servlet that accepts database requests using HTTP/HTTPS protocols and displays the results in XML. The EDG's authentication and authorization software, edg-java-security, analyzes the user's rights to execute operations based on the user certificate presented to the system.

In edg-java-security, the authentication is based on a hand-shaking protocol in Secure Sockets Layer and Transport Layer Security. The server and client send each other their X509-format certificates and messages encrypted by their private keys that are related to the public keys included in the X509 certificates. This way the server and the client authenticate themselves as owners of their respective certificates (Security Coordination Group, 2003).

When a proxy certificate is used as a credential, the user sends their certificate and the proxy certificate to the server (Foster and Kesselman, 1997). The proxy is signed by the user and the user's certificate is used to verify the proxy's signature. This way the chain of trust is delegated to the proxy. The proxy certificate can be used by the user for access to various services, as it carries the user's signature as identification. However, the only apparent feature of the proxy is its issuer, i.e. the user's certificate subject like "O=Grid, O=NorduGrid, OU=hip.fi, CN=Joe User". Each Grid service needs to decide independently the access rights of each certificate owner. This can be improved by introducing extensions to the certificate; in our case a VO extension that states the user's VOs and their role in each of them.

3 AUTHORISATION AND SECURITY IN RELATIONAL DATABASES

Authorisation and security are essential in client-server database systems. In this section we discuss briefly how they are implemented in SQL databases. We follow the book by Elmastri and Navathe (Elmasri and Navathe, 1994).

In general, two methods are used in database access control: discretionary and mandatory access control. In discretionary access control different privileges for database objects (e.g. tables = relations, columns = attributes) are granted to the users while in mandatory access control the data and the users are classified in different security classes. A user, in order to view the data, must have the same or higher security class than the data in question. In the following discussion, we will refer to discretionary access control methods since almost all relational database systems use it while the mandatory control method is used only in some special systems. Moreover, the SQL standards support only discretionary access control.

In SQL, privileges can be assigned to the account (user) level or the database object (relation) level. At the account level the privileges define what operations a particular user can perform in general, and in the database object level the privileges specify the operations a user can perform on the object (e.g. select, modify). In order to perform an operation, the user must have both account level and the object level privileges.

The basic privileges for relations (tables) are *select*, *modify*, and *reference*. The select privilege allows the user to retrieve data from the relation and is defined only on the relation level; views can be used to allow only some attributes to be retrieved. The modify privilege allows the user to modify the data and can be defined in a more detailed manner as *update*, *delete*, and *insert* privileges. The modify privilege is also defined on the relation level, and the update and insert privileges can also be given on the attribute (column) level. The reference privilege allows the user to define references to a relation, e.g foreign key constraints.

SQL has *grant* and *revoke* commands for defining privileges. With the *grant option*, a privilege can be given to the user so that he can grant it further. SQL also supports roles. The role is "a set of privileges" that can be assigned to the user. This makes administration easier since several privileges do not need to grant separately to the user.

4 VOMS

Essentially the Virtual Oraganization Membership System (VOMS) presents an extension to a user's X509 proxy certificate, that includes their VO membership information. When a VOMS-proxy is generated with the `voms-proxy-init` command is used, the VOMS server is contacted to request a VOMS-extended proxy certificate that follows the

standard X509v3 (Housley et al., 1999) certificate format. All the standard fields of the proxy certificate are used to store the user's authentication information. The X509v3 extension (1.3.6.1.4.1.8005.100.100.1) part is used to include authorization information in the user's proxy certificate. The authorization information is stored in triplets of GROUP, ROLE and CAP (special capability rules, like disk space assignments).

When a user wishes to use a grid service, they pass their credential (in this case the VOMS-extended proxy certificate) to the service interface, for example, Spitfire. The grid service then extracts the user's authorization information from the extension part. The extensions part of the proxy certificate also includes the VOMS signature and validity period of the role mappings. The VOMS signature is used to to verify that a trusted VOMS service has attached the authorization information to the user's proxy certificate.

The actual information that the VOMS extension contains can be configured by the VO's administration, by using VOMS'es user interface.

In the future, it is foreseen to use Attribute Certificates (Farrell and Housley, 2002) to store the authorization information instead of extending the proxy certificate. This will not change the way information is used in a service since the same information is extracted from a different credential passed by a user.

5 COMBINING VOMS AND SPITFIRE USER AUTHORIZATION MECHANISMS

In the Spitfire service, database roles are divided into categories of "read", "write", "update", and "create" by default. A user that has been assigned the read role is allowed to browse the database in question; similarly, "write" role owner can insert data, "update" role owner can modify data, and "create" role owner can create new tables. Normally, of course, a user can have many roles; for instance in the case of a database administrator all of the above.

A *policy* is a collection of mappings of roles and users. Policies are designed by EDG's TrustManager software components. For instance, policy "default" can be based on a regular expression mapping that allows roles "read", "write", "update", and "create" to a person whose certificate subject is "O=Grid, O=NorduGrid, OU=hip.fi, CN=Joe User", and "read" to persons whose certificate subject contains "O=Grid, O=NorduGrid, OU=hip.fi".

Spitfire can access any relational database with a Java connector, but the default implementation is based on MySQL (MySQL AB, 2001). With MySQL, the roles are applied on database-wide basis [1]. Different VO's can have their databases accessed through a single Spitfire service and the access configuration is managed with standard Java database access methods.

6 USE CASE: ACCESSING SPITFIRE WITH A CLIENT PROGRAM

Spitfire offers standalone Java and c++ clients for accessing the service. A use case is presented to illustrate database access with VOMS certificates.

First the user requests a VOMS certificate with `edg-voms-proxy-init` command. The certificate is written and stored to a file "/tmp/x509up_uid" (where id is the user's identity number in the computer). When the client accesses a service, the VOMS (proxy) certificate is sent to the service. The service is then able to extract the authorization information from the certificate.

The service finds out which VO the user belongs to. Further, the role of the user in his VO is also extracted from the certificate. With this information the service is able to define the user's access rights. This removes the need from a service to know specific user ids and their mapping to database access rights.

In the case of successful authorization, the result of the user's query is returned to the client. An extract of sample client code is shown in Figure 1, where the programmer requests an XML output of query "select all columns from table repcat in database GRID such that the column LFN contains cms.cern.ch". The security and authentication features are completely invisible to the programmer.

There, we notice that the user requests policy "voms-based" and and uses his proxy certificate named "/tmp/x509up_u6385".

7 CONCLUSIONS

In this paper, we have presented a design and an implementation of virtual organizations and database access. The main benefits of the design are that it separates authentication and role authorization in manageable modules; and that here is no need to enter user id - password -pairs because of certificates. Using VOMS adds the benefit of easy administration; the VOMS admin can define global policies how to

[1] MySQL's "database" rougly corresponds to Oracle's tablespace. Many database vendors, including MySQL and Oracle, implement at least "per database" and "per table" access control.

```
public final class SimpleQuery {
    public static void main (String args[]) throws Exception {
        String endpoint = null;    endpoint = args[0];
        SpitfireBaseServiceLocator sfLocator = new SpitfireBaseServiceLocator();
        SpitfireBase sfBase = sfLocator.getSpitfireBase(new java.net.URL( endpoint ));
        try {
            System.out.println("The result as XML\n");
            String c = "LFN like \"%cms.cern.ch%\"";
            System.out.println(sfBase.simpleSelectAllAsXML("GRID","repcat",c, 0));
        } catch (Exception x) {
            System.out.println(" ERROR: " + x.getMessage() );
        }
    }
}
```

```
java -classpath [...]
-DgridProxyFile=/tmp/x509up_u6385
-Dedg-security.policy=voms-based
SimpleQuery https://localhost:8443/Spitfire/services/SpitfireBase

The result as XML
<?xml version="1.0" encoding="UTF-8"?>
<ROWSET><ROW num="1"><LFN>lfn://cms.cern.ch/dataset/file001</LFN>
<PFN>pfn://cms005.cern.ch/data/hh01/file001</PFN></ROW>
[..]
<ROW num="5"><LFN>lfn://cms.cern.ch/dataset/file005</LFN>
<PFN>pfn://cms005.cern.ch/data/hh01/file005</PFN></ROW></ROWSET>
```

Figure 1: Sample code, a command to invoke it, and its output.

access database services within a VO and thus ease local administration procedures.

As discussed in Section 5, the system can only be accessed using specific client programs, not for instance a web browser. However, we are in process of incorporating VOMS functionality in GridBlocks Agent software (Mobile Analyzer, see (Karppinen et al., 2003)) that enables simultaneous queries to distributed databases and has a graphical user interface.

EDG software is open source, and available at http://datagrid.in2p3.fr. GridBlocks is open source software available at http://gridblocks.sourceforge.net.

REFERENCES

Alfieri, R., Cecchini, R., Ciashini, V., dell'Agnello, L., Frohner, A., Lorentey, K., and Spataro, F. (2003). VOMS an authorization system for virtual organizations. In *Proceedings of the 1st European Across Grids Conference - Santiago de Compostela, Spain, 13-14 February 2003*.

Elmasri, R. and Navathe, S. B. (1994). *Fundamentals of database systems (2nd ed)*. Benjamin / Cummings.

Farrell, S. and Housley, R. (2002). Rfc 3281, an internet attribute certificate profile for authorization. Available on http://www.ietf.org/rfc/rfc3281.txt.

Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2).

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3).

Housley, R., Ford, W., Polk, W., and Solo, D. (1999). Rfc 2459, internet x.509 public key infrastructure certificate and crl profile. Available on http://www.ietf.org/rfc/rfc2459.txt.

Karppinen, J., Niemi, T., and Niinimaki, M. (2003). Mobile analyzer - new concept for next generation of distributed computing. The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGrid 2003), Japan, May 2003. Available on http://ccgrid2003.apgrid.org/online_posters.

MySQL AB (2001). Mysql. Available on: http://www.mysql.org.

OgsaDai (2002). Open grid services architecture data access and integration. Available on: http://www.ogsadai.org.

Project Spitfire (2001). Project Spitfire. Available on: http://spitfire.web.cern.ch.

Security Coordination Group (2003). Datagrid security design. Technical report, European DataGrid Project.