VOMS Architecture

v1.1

05/09/02

1. The VOMS System

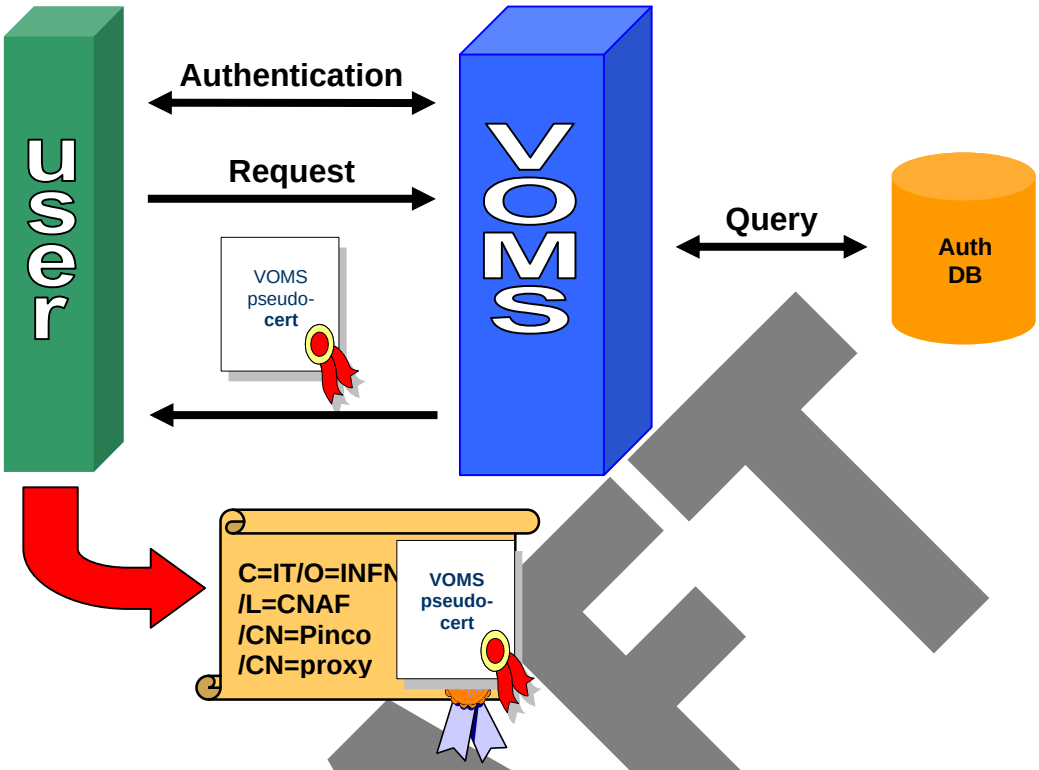
The VOMS System is composed by the following parts:

- **Server**: receives requests from a client and returns information about the user.
- **User Client**: contacts the server presenting a user's certificate (or possibly proxy) and obtains a list of groups, roles and capabilities of the user.
- **Administration Interface**: used by the VO administrators (adding users, creating new groups, changing roles, etc...).

1.1 Operations

One strong requirement we have is to disrupt as little as possible - from the user's standpoint - the creation of the user proxy certificate. To achieve this we have modified the "*grid-proxy-init*" command (referred hereinafter as "*voms-proxy-init*") in order to get, first, the info from the VOMS server. The info is returned to the user in a structure containing also the credentials both of the user and of the VOMS server and the validity; all these data are signed by the VOMS server itself. We call this structure a "Pseudo-Certificate"[1]. The user may contact any VOMS as he needs. The modified procedure creates then the user proxy certificate inserting these data into it in a non critical extension. In this way, the variations in the new format of user proxy certificate should be transparent to the old programs, thus allowing a smooth transition to the new system.

[1] The reason why we don't use a "true" Attribute certificate [3] is mainly to be found in the limitations on the length of the certificate due to the interaction between OpenSSL and Globus. We try to keep the size of these pseudo-certificates to the minimum, so as to minimize the risk to end up with a proxy too big.



More in details, the *voms-proxy-init* command, which produces a "VOMS-aware" proxy certificate, can be divided into the following points:

1. Client and server mutually authenticate themselves and establish a secure communication channel using standard Globus API..
2. The Client sends the request to the Server.
3. The Server checks the correctness of the request and sends back the required info (signed by itself): User Info Pseudo-Certificate (PC).
4. The Client checks the validity of the info received.
5. Steps 1—4 are repeated for each Server the Client needs to contact.
6. The Client creates a proxy certificates with an extension (non critical) containing all the info received from the contacted VOMS Servers (as separate PC's).

1.2 Format of User Info Pseudo-Certificate

The signed info returned by the VOMS Server to the Client is composed by the following fields.

- **Holder**
  Subject and Issuer of the certificate of the user requesting the info.
- **Issuer**
  Subject and Issuer of the VOMS server certificate.
- **Validity**

  - **notBeforeTime**
  - **notAfterTime**

    For the moment it is an unique value for all the VOMS: if necessary it could become different according to the kind of info returned. Of course, the final decision is left to the local site.
- **VO name**
- **Attributes**

  The attributes (groups and roles) are returned as two lists. See Appendix C (when it will be ready!) for details.
- **Signature**

  VOMS Signature of the above data.

2. Modifications to Globus

2.1 Resource Broker

2.2 Gatekeeper

The *Gatekeeper*, in addition to normal certificate checking, has to perform all the operations that the client performed on the pseudo-certificates if it wants to use their information. This can be easily done with an ad hoc LCAS plug-in.

However, as the VOMS info are included in a non critical extension of the proxy certificate, this can be used even by "VOMS-unaware" *Gatekeepers*, thus maintaining compatibility with previous releases.

2.3 Security Considerations

The VOMS server does not add any security issues at user level since it performs the usual GSI security controls on the user's certificate before granting rights (it must be signed by a "trusted" CA, be valid and not revoked).

On the other hand, even compromising the VOMS server itself would be not enough to grant illegal access to resources since the authorization data must be inserted in a user proxy certificate (i.e. countersigned by the user himself). Hence the only possible large scale vulnerabilities are denial of service attacks (e.g. to prevent VO users to get their authorization credentials).

The main security issue about proxy certificates is the lack of a revocation mechanism; on the other hand these certificates have short lifetimes (12 hours, typically). For a detailed discussion about security implications of restricted proxy certificates see [12].

3. Server Architecture

3.1 Introduction

These are the differences between the v0 version (released 31 July) and v1.1 (in collaboration with CERN)

- there aren't user's capabilities (e.g. ACL's) and queries (**capability** and **queries** tables);
- it is possible to have more than one administrator. Their privileges are specified in the **acl** table;
- every transaction is fully traced.

3.2 Database

The server is essentially a front-end to an RDBM, where all information about user are kept. It accepts the following requests in text format:

A	Returns all info regarding the user.
B <*group*>:<*role*>	Returns all info regarding the user as a member of the specified group with the specified role
G <*group*>	Returns all info regarding the user as a member of the specified group
L	Returns the list of all the available queries
R <*role*>	Returns all info regarding the role specified

Note: All the queries have an implicit <*userid*> field, derived from the certificate presented by the user to the server.

The option **`-print`** allows to display the result of the query, instead of generating the proxy certificate.

3.3 Tables

The structure of the tables is the following: (please note that type information has been left deliberately vague, because different RDBM's may need to declare the same field with different types).

Primary key fields are shaded.

m

user	user's identifier	number
group	user's group identifier	number

role	user's role identifier	number
createdBy		number
createdSerial		number

users

uid	user's identifier	number
dn	DN of the user's certificate (**Subject**)	text
caid	Identifier of the user's CA	number
cn	User's CN	text
mail	User's e-mail	text
cauri	URI of the user's certificate (of CA LDAP server)	text
createdBy		number
createdSerial		number

ca

caid	Certification Authority identifier	number
cadn	CA Subject	text
cadescr	Name of Certification Authority	text

This is the only table without the traceability fields.

acl

aclid	acl identifier	number
principal	DN of administrator's certificate	number
operation		number
allow		boolean
createdBy		number
createdSerial		number

The table contains the DN's of the administrators' certificates, together with their permissions.

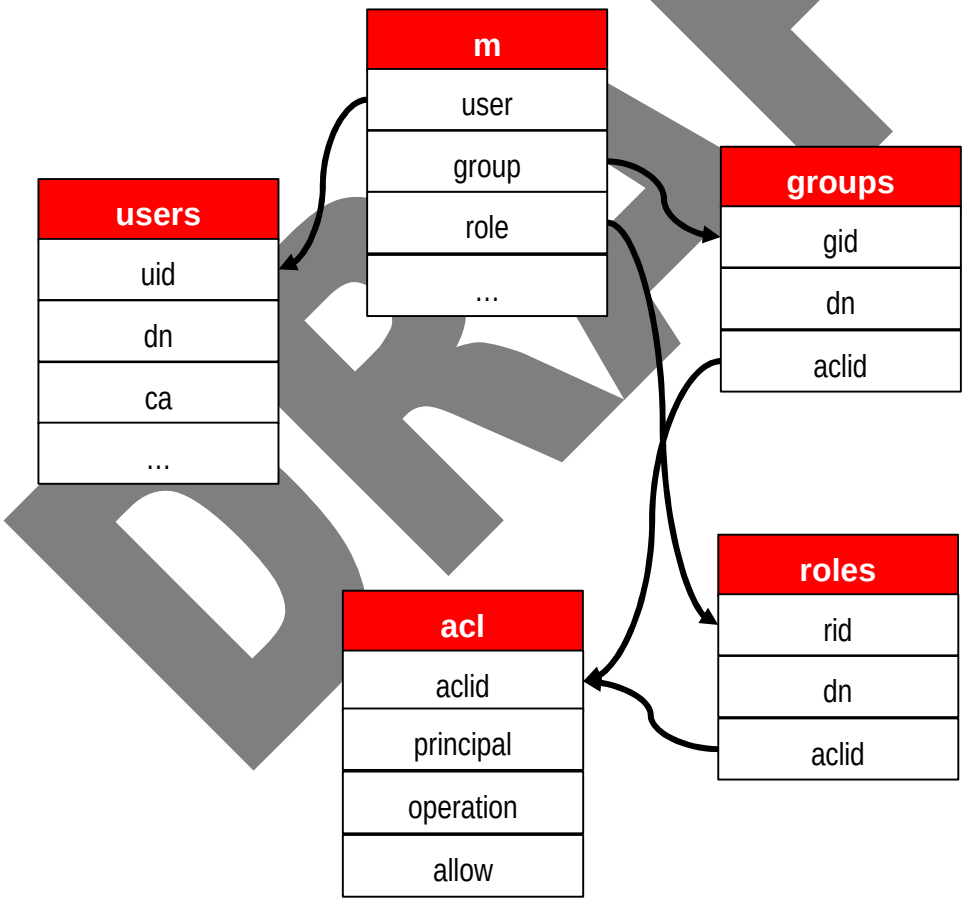
groups

gid	group identifier	number

dn	group name	text
aclid	the acl identifier of the administrator	number
createdBy		number
createdSerial		number

roles

rid	role identifier	number
dn	role name	text
aclid	the acl identifier of the administrator	number
createdBy		number
createdSerial		number



3.4 Traceability

Every table has two fields (createdBy and createdSerial) which are filled at every update operation. Deleted and modified rows are kept in another database.

3.5 Replicas

References

[1] Foster, I. and C. Kesselman, eds. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann (1999).

[2] R. Housley, T. Polk, W. Ford and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC3280 (2002).

[3] S. Farrel and R. Housley, *An Internet Attribute Certificate Profile for Authorization*, RFC3281 (2002).

[4] *Architectural design and evaluation criteria: WP4 Fabric Management*, **DataGrid-04-D4.2- 0119-2-1** (2001).

[5] *The DataGrid Project*: http://www.edg.org/

[6] *The DataTAG Project*: http://www.datatag.org/

[7] *iVDGL - International Virtual Data Grid Laboratory:* http://www.ivdgl.org/

[8] *The Globus Project*: http://www.globus.org/

[9] *Grid Security Infrastructure*: http://www.globus.org/security/

[10] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid*, International Journal of High performance Computing Applications, **15**, 3 (2001).

[11] L. Pearlman, V. Welch, I. Foster, K. Kesselman and S. Tuecke, *A Community Authorization Service for Group Collaboration*, IEEE Workshop on Policies for Distributed Systems and Networks (2002).

[12] S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman and C. Kesselman, *Internet X.509 Public Key Infrastructure Proxy Certificate Profile*, draft-ggf-gsi-proxy-04 (2002).

[13] J. Vollbrecht et al, AAA *Authorization Framework*, RFC2904 (2000).

A. Sample “VOMS-enabled” Proxy Certificate

This is a sample proxy certificate with pseudo-certificates from two VOMS.

B. API's

This is a very preliminary list.

createUser (dn, ca)
deleteUser (dn)
createGroup (name)
deleteGroup (name)
createRole (name)
deleteRole (name)
grantAttribute (user, group, role)
removeAttribute (user, group, role)
getAttributes (user, group, role)
listMembers (group)
addACLEntry (attribute, allow, principal, op)
removeACLEntry (attribute, principal, op)
setACL(attribute, {principal, op, allow})

C. ASN.1 definition