VOMS-aware Middleware Deployment : Assessment of Loose Ends *

Oscar Koeroo, Ronald Starink, Jeff Templon

June 27, 2007

Contents

Recent deployment of VOMS-aware middleware components such as the gLite WMS and `lcg-info-dynamic-scheduler` have uncovered several undefined (or equivalently, "multiply-defined" constructs related to VOMS FQANs. This document attempts to concisely and clearly state the problem and give some concrete examples.

1 Introduction

The question that the authors would have liked to have answered in this document is, "what do the various constructs found in the CE ACBRs (AccessControlBaseRules)

*$Id : Vdepl - summ.tex, v1.42007/05/1014 : 20 : 54templonExptemplon$

mean?" Unfortunately, the answer appears to be the empty set : there is no well-defined meaning, in practice there are only various (different) conventions.

This situation leads to the following issues:

- it is not clear how to interpret the meaning of an ACBR
- it is not clear how a "match" between FQANs should be defined
- it is not clear to users or WMSes, how they should select resources based on ACBRs
- it is not clear to site administrators, how they should configure YAIM, LCMAPS, the batch system, and the information providers.

An attempt was made by most parties to follow what was understood as the "LCMAPS model", which means that when a user tries to do something, the first FQAN in her proxy gets inspected, the "best match" for this FQAN is found amongst all the possibilities, and this best match gets used.

What was overlooked is that the LCMAPS model involves

- the FQAN syntax and rules for matchmaking
- an algorithm that boils down to a switch statement
- the order of the switch statement (which is the order of the lines in the LCMAPS `groupmapfile`.)

This means that knowing the FQAN syntax and semantics, and FQAN matching rules, is not enough; in order to be able to choose the "right" ACBR for matchmaking, all products must implement the same switch statement as is done on the site in question. Stated otherwise, unless all middleware makes the same assumption about all three of these items, different products will get incompatible matchmaking results. What is particularly nasty is that to really get it right (*assuming we stick with this model*), every site will have to publish, as part of the ACBR construct, a value which indicates the relative order of this ACBR in the site-local `groupmapfile`, or we will have to design a set of ACBRs which ensures that a given FQAN will never match more than one.

2 FQAN and ACBR format

The expression of an access control statement in a VOView is not clearly defined. Consider the following list:

Example set of ACBRs:

1. `VOMS:/atlas`
2. `VOMS:/atlas/*`
3. `VO:atlas`
4. `VOMS:/atlas/Role=NULL/Capability=NULL`
5. `VOMS:/atlas/Role=production/Capability=NULL`

6. `VOMS:/atlas/*/Role=production/Capability=NULL`

There are a few problems with these items in the list. Row number one states an incomplete FQAN and thus invalid FQAN. The string `/atlas` might mean the equivalent of `/atlas/Role=NULL` or `/atlas/*`. In the first case, only the base VOMS group can match that line. In the latter, any FQAN belonging to the ATLAS tree can match.

3 FQAN matching

Thus for the first line, it's unclear whether this would only match `/atlas/Role=Null/Capability=Null`, or whether it would match any FQAN (unlimited groups and/or roles/capabilities) as long as it begins with `atlas`.

The second row expresses a very clear wildcard statement. This line will match all FQANs that start with `/atlas/`. The star character is a wildcard that can match all the possible characters that are allowed by the VOMS system (explained in the document VOMS Memo version 10, located at OGF OGSA AuthZ docs online).

Row number three expresses the old-style (pre-VOMS) notation. This has always meant that it will match everybody in the ATLAS VO, regardless of whether VOMS credentials (or old-style grid proxies) have been used. In reality this is not the whole story. The next section will describe this problem in more detail.

The fourth, fifth and sixth row are also very clear in what they express. The fourth and fifth row require an exact match of the FQAN; the sixth line can only be matched by an FQAN that starts with `/atlas/` and has a `/Role=production` part in it (the group part is not constrained). For one of the authors, even this FQAN is not clear; does it match if the FQAN being presented as a candidate is

`/atlas/soft-valid/Role=lcgadmin/Role=production/Capability=NULL`?

With this in mind, what should we get if we match the following single FQAN

`/atlas/Role=production/Capability=NULL`

against the list?

This FQAN must match with lines 2, 5 and 6. Line 3 should match as well, if we decide that VO:atlas really means "everything in ATLAS"; line 1 will or will not match according to whether one believes in the "implicit wildcard".

It will never match line 4.

4 Use Cases

Examples from various systems of what actually happens in practice with the VOMS or VO information.

4.1 Storage

How storage elements treat a proxy, should depend on the operation that is performed. Furthermore, since there are various storage element types (Classic SE, DPM, d-Cache, Castor), each of them may have its own implementation.

It is not clear how the various storage elements deal with VOMS proxies at this moment. As a first step, this should be investigated in more detail. On the longer term,

all storage elements should follow the same mapping rules, in order to make it predictable for users, VO managers and system administrators where data will be written (i.e., how much space should be allocated) and which users (or better, VOMS proxies) will have access to read that data.

4.1.1 Writes

When a storage element receives a put request, it should only consider the primary VOMS role for the mapping. This is needed to guarantee consistency in the location (e.g. a disk pool or file system) where the data is stored. That is not only relevant to manage the available disk space for the various groups and roles, but also to prevent pollution of storage space.

Although the following example is somewhat artifical, it clearly indicates why only the primary VOMS role should be considered for write operations. Suppose a user is member of 2 VOs, say Atlas and LHC-b. For both VOs, she is just a regular user without membership of any special groups (or roles). Let's assume that her primary VOMS role is Atlas, implying that she wants to do Atlas work. Now she wants to store data on a storage that only supports LHC-b. If the SE would take any other than the primary role into account, the Atlas data would be stored in the LHC-b disk storage, and would consequently be accounted to LHC-b. The same situation could occur if the storage element would support Atlas, but there is no space available for Atlas.

Note that the above example is also valid if the user is not member of two VOs, but of two different groups in the VO. For instance, a user could have the following VOMS groups and roles in her proxy: /VO=Atlas/NL/Role=Null and /VO=Atlas/Role=production. It would still be undesirable to mix data from the generic Atlas production with data from local analysis.

4.1.2 Reads

For read operations, all VOMS roles and attributes should be considered. If the user has permission to read data through the presence of a certain VOMS attribute, then that should be sufficient. It is not particularly important if that corresponds to the primary role at the moment of working. Reading is after all an operation that does not modify the contents of the data or affect the available space on the storage element.

4.2 Workload Management Systems

The workload management systems (WMS), i.e. the lcg-RB and the glite-WMS, consult the information system to determine which grid sites match the requirements for a given job. The job requirements may be specified by the user, but part of them are determined by the VOMS roles and groups in the grid proxy. During the matching process, the workload management systems filter all grid sites that satisfy the requirements for the job. That results in a list of all queues, spread over all accessible computing elements, to which the job could be submitted.

The final step in determining to which site the job should be submitted, is to evaluate the ranking. The user may specify what criterium to use for ranking. By default, it is the estimated response time (ERT), i.e. the estimated time after which the job could run, attempting to minimize the waiting time before the job starts to run. The ERT

is published in the information system in a view. Each view has its associated Access Control Base Rule (ACBR), that determines to which VOMS roles and groups it applies.

Using the default ranking, the workload management system will determine the minimum ERT from all views to which the user got access. Once the view with the minimum ERT is chosen, the WMS will redirect the job to the computing element and queue associated to that view. It is important to note, that if there are multiple matches for one computing element, the selection by the WMS does not necessarily correspond to the actual ERT on the site; more details on this are found in the section on Scheduling.

4.3 Scheduling

For the `dynamic-scheduler` program, various data are published (job counts, estimated response times) in various views, to which ACBRs are attached. The current implementation assumes that a single ACBR is attached (not counting DENY tags which are currently ignored by the program).

In practice, the information provider has a map (`vomap` in the program's config file) which maps unix groups onto specific FQANs. It is possible to map multiple groups onto a specific FQAN, but in the current implementation it is not possible to map a single unix group onto multiple FQANs. In the best of all worlds, the FQAN to group mapping in the config file would be the same as that in the LCMAPS group map file.

The concept of "inclusive" matches is well-defined in the information provider; it consistently makes the most pessimistic assumption. If multiple unix groups are mapped to a single construct like "VO:lhcb", then the estimate printed by the program is the worst estimate amongst all the mapped groups. The rationale is that the site is *publishing* this information to the outside world and not in control of how it will be used; it is better from an application point of view to have your wait be shorter than expected, so the longest estimate is printed.

To illustrate how this should work, suppose we have

- /lhcb/prd mapped to group lhcbpr
- /lhcb/Role=lcgadmin mapped to group lhcbsgm
- the rest of lhcb (via LCMAPS switch statement) mapped to group lhcb.

Suppose we have the following info for the unix groups:

- group lhcb : 27 running jobs, ERT 543 sec
- group lhcbsgm : 3 running jobs, ERT 23 sec
- group lhcbpr : 49 running jobs, ERT 1234 sec

Then we should see the following:

```
View 1:
VO:lhcb
RunningJobs : 79
ERT: 1234 sec
```

```
View 2:
VO:lhcb
DENY:/lhcb/prd
RunningJobs: 30
ERT: 543 sec

View 3:
VOMS:/lhcb/prd
RunningJobs: 49
ERT: 1234 sec

View 4:
VO:lhcb
```

In order to print this information, some modification is needed to the program, not because of the "inclusive" concept being absent, but because the assumption "single unix group only mappped to one single ACBR" is built into the program's input stage.

4.4 Execution

At the computing element, all mapping is handled by LCAS/LCMAPS. From the perspective of the system administrator, there is a well-defined order of the mapping. However, the mapping is not published. For the outside world, meaning both users who want to create a VOMS proxy with a certain primary role as well as workload management systems, that implies that it is not a priori clear how the mapping will happen. This is particularly true if the information system contains more than one view with an ACBR that matches for the VOMS proxy.

An operating system can't use grid credentials directly. The system will only be able to work with its own numerical representation for users and groups. On Unix(-like) systems these numerical representations can be split into three different types:

- Unix User ID (UID)
- Unix Group ID (GID or PGID)
- Secondary Unix Group ID (SGID)

An executing process has variaty of different attributes. Here we need to focus on the attributes that hold the UID, PGID and SGID of a process. In reality there is a difference between the real and effective UID, PGID and SGID, but for the ease of the explenation we'd like to pretent that there is no such difference.

When a user submits a job to the Grid it will eventually end up at a compute cluster. This is done through the submission to a globus gatekeeper. With a focus set on the LCG-CE this will mean that the job has been submitted to the edg-gatekeeper. The edg-gatekeeper difference from the orginial globus 2.4 gatekeeper in the effect that it will call an external Authorization and Mapping framework.

The Local Center Authorization Service (LCAS) framework is an authorization framework that can execute various plugins that will examine the jobs properties and the user's

used credentials for this job. If the user is authorized based upon his provided credentials then the Local Credential MAPping Service (LCMAPS) will take over. LCMAPS will call various plugins that each have a specific task to acquire information from the provided credentials, acquire information from the local system and result in a mapping. In the LCMAPS sense: mapping is to make a Unix representation of all the provided credentials.

LCMAPS uses a bunch of files but I'd like to focus on the grid-mapfile and groupmapfile because these files describe how LCMAPS should map a given set of credentials to Unix credentials.

[...grid-mapfile...]

```
"/O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo" .mypool
"/atlas/Role=production/Capability=NULL" .atlb
"/atlas/Role=NULL/Capability=NULL" .atlas
"/atlas/*" .atlas
```

[...groupmapfile...]

```
"/atlas/Role=production/Capability=NULL" atlb
"/atlas/Role=NULL/Capability=NULL" atlas
"/atlas/*" atlas
```

[...voms-proxy-info example (shortend)...]

```
/atlas/Role=production/Capability=NULL
/atlas/Role=NULL/Capability=NULL
/atlas/somespecialgroup/Role=NULL/Capability=NULL
```

The grid-mapfile can be used in multi-mode mixing DNs and FQANs. The plugins of LCMAPS will nicely keep the difference between them, but effectively there is a difference in how this works. The DNs and FQANs can be mixed up through the file, but since the grid-mapfile and groupmapfile are read from top to bottom the list must be ordered in a "most specific FQAN first". LCMAPS will not order it prior to its use to avoid 'magical' (read: hard to reproduce inside the human mind) mappings.

When VOMS isn't used only the DN will be considered for mapping and only the grid-mapfile will be read. The first full string match of that DN is used and the localaccount or the poolaccount is selected. In the example grid-mapfile the DN of Oscar will map to the pool called mypool, resulting in an account like `mypool001`. This method only selected to account `mypool001`. To be able to fully enforce this account into the properties of the current process of the edg-gatekeeper the active LCMAPS plugin will also acquire the PGID and optional SGID from the system. It depends on the system configuration if the password-file, LDAP directoy or something else is utilized for this information completion. When the UID and PGID is known the final stage of LCMAPS can be called to enforce these acquired Unix credentials into the properties of the process. In the case of the edg-gatekeeper the process is running with root priviledges. These priviledges will be squashed after the enforcement of the mapping. If they can't be squashed then that is a failure condition. Nobody should be able to submit a job and execute with local root priviledges.

When VOMS is used to submit a job, then a lot of routines are used in the same way as without VOMS. But now we'll need two files. The first file is the grid-mapfile

that describes how VOMS FQANs must be mapped to a specific localaccount[1] or a pool of accounts. The second file is the groupmapfile. That file contains the same type of information but to be used to determain the Unix group on to wich the grid credentials (the FQANs) need to be mapped. The groupmapfile can describe how to map an FQAN to a (regular) localgroup or a pool of groups. The poolgroups feature is not used in production so I will not describe it here to avoid overcomplexity.

First the Unix Group IDs will be acquired from the FQANs. Given the attributes from the voms-proxy-info example and the given groupmapfile file will LCMAPS try to map each FQAN to a Unix Group ID. First the FQAN `/atlas/Role=production/Capability=NULL` will be evaluated. This FQAN has an explicit mapping and will result into a mapping to the Unix Group ID `atlb`. The numerical representation of the Unix Group is stored into the LCMAPS framework. This first found and mapped FQAN will result into the Primary Group ID of the process. This is important since most of the cluster schedulers can only evaluate a process upon the Unix User ID and Unix Group ID. Since the UID can be any in the pool and the GID is actually telling something more about the group affiliation of the user, that Unix-type credential will be used to make a scheduling descision (more about that later in the document). All the FQANs will result into Secondary Unix Group IDs. The second FQAN in the list to evaluate is `/atlas/Role=NULL/Capability=NULL`. In the example groupmapfile we can see a specific mapping of this FQAN and it will result into a mapping to the Unix Group called `atlas`. The numerical representation will be stored into the LCMAPS framework. The last FQAN `/atlas/somespecialgroup/Role=NULL/Capability=NULL` does not have a specific mapping. It does match the wildcarded `/atlas/*` line in the groupmapfile. Therefor the resulting mapping is the Unix Group ID for `atlas` again.

Effectively the acquired groups atlb, atlas and atlas could have been named differently and still be using the same Group ID numerically. That's why we've chosen to store the acquired numerical representation only.

In the next stage LCMAPS will launch a plugin to acquire the Unix User ID regarding the grid-mapfile and the credentials of the user. In this VOMS specific case the poolaccount selection will be triggered not by the DN but by the first FQAN in the list of provided FQANs. In this case the FQAN `/atlas/Role=production/Capability=NULL` will map to the pool `atlb` which can result in the Unix account `atlb001`.

Since that in the final stage a poolaccount is selected from a specific pool instead of the statically acquirable groups, the gridmapdir mapping between the DN and the poolaccount will be done. With the regular poolaccount hardlinking to a URL encoded string in the designated gridmapdir will the VOMS variant of this perform the same kind of hardlinking to the URL encoded DN and the numerically respresented set of GIDs. These GIDs are ordered PGID first, then SGIDs and the SGIDs are ordered numerically (from low to high numbers). This is done so because users might be mapped to the same effective Unix credentials and may thus return to the same poolaccount. If the set of FQANs differs a bit and the mapping to Unix credentials results into a different set of credentials then this user should be mapped to a new account. The granularity can be configured. We advise at least to have the account change on an alteration of the first two Group IDs. Each numerical GID can only be listed once because having two or more times the same SGID enforced is nonsence.

[1]Mapping VOMS FQANs to one specific local Unix account is regarded dangerous for possible legal implication because your mapping a group of people into one account and thus dismissing auditability.

At the moment all the production sites are configured in such a way that users are able to failover from the VOMS mapping plugins (when they fail or don't have VOMS attributes) in to the old style poolaccount mapping plugins. This will succeed if old style DNs are listed in the grid-mapfile for that plugin.

4.5 Accounting

5 Technical Decisions

This segment is lifted directly from the EGEE Job Priorities conclusions document. We are inclined to believe that the wrong choice was made in picking "exclusive" reporting, as it yet again introduces entanglement : it's not clear what VO:atlas means at a site, unless you know the entire list of ATLAS views at the site, and even this may not be enough.

This segment is included from the report of the EGEE Job Priorities Working Group to document certain deployment choices. An example is the syntax with which YAIM should print the VOMS FQANs.

A number of implementation choices need to be made regarding the entire scheme. Here we list those made so far. These are meant to be pragmatic — we don't know enough to make a definitive statement, but we do know that we want to begin quickly. These choices are hence weighted towards easy implementation and need to be reviewed once we have sufficient experience.

5.1 Syntax of ACBRs

All code that needs to parse the AccessControlBaseRules should conform to the following syntax:

```
GlueCEAccessControlBaseRule:<OWS><SNC>:<SNC>
```

where `<SNC>` means "string containing no colons" and `<OWS>` means "optional whitespace". Whitespace is forbidden between the `GlueCEAccessControlBaseRule` tag and the semicolon that follows, but in general is OK elsewhere. Here are examples of "good" syntax:

```
GlueCEAccessControlBaseRule: VO:atlas
```

and similarly with VOMS stuff. What is *not* OK is

```
GlueCEAccessControlBaseRule:   VO :atlas
GlueCEAccessControlBaseRule:   VO: atlas
GlueCEAccessControlBaseRule:   VO : cms
```

due to spaces around the colons (basically handling spaces requires a rewrite of various classads matching functions scattered throughout the WMS code). The following

```
GlueCEAccessControlBaseRule:   VOMS:/atlas/Role=sys:admin
```

is also not OK because this has a second colon, and then the split becomes ambiguous. Similarly this is also not OK:

```
GlueCEAccessControlBaseRule: SERVICE:CLASS:lhcb_bronze
```

5.2 Reporting by Information Provider

If a certain queue (`GlueCE`) supports both CMS as a whole, as well as a special CMS share, there are choices to be made in how to report this. One might make the choice that "CMS as a whole" should mean to report everything belonging to CMS, including the jobs belonging to the special share supported by the same queue.

We decided to make the reporting exclusive: "VO-as-a-whole" `VOView` blocks should report numbers corresponding to all jobs / shares for that VO that aren't explicitly reported by more specific blocks.

To be concrete, if we have `VO : atlas` and `VOMS: /atlas/Role=production` supported by a single CE, in the `VOView` block for the production role, job counts and response-time estimates are reported specifically for the production role; for the `VO : atlas VOView`, counts and estimates are reported for all of atlas *except* the production role.

Note there is some work to be done here in matchmaking, since you might have a VOMS proxy that matches more than one of the published FQANs. We need to develop a matching-precedence hierarchy for the long term so that the concept "most specific match" is meaningful.