Journal of Physics: Conference Series **119** (2008) 062032 doi:10.1088/1742-6596/119/6/062032

gLExec: gluing grid computing to the Unix world

David Groep[1], **Oscar Koeroo**[1], **Gerben Venekamp**[1]

[1]Nikhef, P.O. Box 41882, NL 1009 DB Amsterdam, The Netherlands

E-mail: grid-mw-security@nikhef.nl

Abstract.

The majority of compute resources in todays scientific grids are based on Unix and Unix-like operating systems. In this world, user and user-group management are based around the concepts of a numeric 'user ID' and 'group ID' that are local to the resource. In contrast, grid concepts of user and group management are centered around globally assigned identifiers and VO membership, structures that are independent of any specific resource. At the fabric boundary, these 'grid identities' have to be translated to Unix user IDs. New job submission methodologies, such as job-execution web services, community-deployed local schedulers, and the late binding of user jobs in a grid-wide overlay network of 'pilot jobs', push this fabric boundary ever further down into the resource. gLExec, a light-weight (and thereby auditable) credential mapping and authorization system, addresses these issues. It can be run both on fabric boundary, as part of an execution web service, and on the worker node in a late-binding scenario. In this contribution we describe the rationale for gLExec, how it interacts with the site authorization and credential mapping frameworks such as LCAS, LCMAPS and GUMS, and how it can be used to improve site control and traceability in a pilot-job system.

1. Introduction

Inter-organizational and cross-domain Grid computing aims to facilitate coordinated resource sharing by providing a uniform interface to systems managed by independent administrative entities [1]. In order to identify users and user communities, many of the scientific Grid infrastructures today use Public Key Infrastructure (PKI) technologies where both users and the services they access mutually authenticate each other based on X.509 [2] public key certificates. To facilitate single sign-on and delegation, most of the clients use the derived *proxy certificates* [3] to authenticate. These proxy certificates may additionally carry attributes that express user-community membership (*Virtual Organization* or VO membership, in the context of this paper), where such attributes are cryptographically signed by an attribute authority representing the community, e.g. as used in the VOMS [4] system. Together, these comprise a 'grid identity'.

Local services, in particular computing services offered on Unix [5] and Unix-like platforms, use a different native representation of the user and group concepts. In the Unix domain, these are expressed as (numeric) identifiers, where each user is assigned a user identifier (uid) and one or more group identifiers (gid). At any one time, a single gid will be the 'primary' gid (pgid) of a particular process. This pgid is initially used for group-level process (and batch system) accounting. The uid and gid representation is local to each administrative domain.

gLExec is a program to make the required mapping between the grid world and the Unix notion of users and groups, and has the capacity to enforce that mapping by modifying the uid and gids of gLExec and any processes spawned through gLExec. For a service running under a

'generic' uid, such as a web services container, it provides the way to escape from this container uid. It may be used similarly by externally managed services run on a site's edge, such as VO-administered schedulers and job submission systems running at a site as *edge services*. Lastly, in a late-binding scenario, the identity of the target job owner (i.e. the identity of the user that submitted a specific work load) can be declared and set at the instant the job starts executing.

In this paper we describe the use cases that necessitated the development of gLExec, the implementation boundary conditions, and the actual implementation of gLExec with the supporting authorization and credential mapping frameworks. Policy issues that influence the deployment of gLExec are discussed.

2. Rationale for gLExec

Translating grid identities to local Unix identities has usually been performed at the fabric boundary. At the time a grid job is submitted for execution to a computing service, the client establishes a mutually authenticated network connection using X.509 credentials [2], and authorization and credential mapping is done immediately, before the job is submitted a local resource management system (LRMS) such as a batch system, and before any client-job management is started. It is the mechanism implemented by, e.g., the pre-web services Globus Toolkit's *globus-gatekeeper* [15]. Various mechanisms today extend this basic functionality, such as introduced via LCAS and LCMAPS [16, 17] in the *edg-gatekeeper* [16] and via GUMS [18] in the PRIMA [19] extensions, but do not alter the basic methodology.

The combination of the *gatekeeper* and a job management system we call a site's Grid Computing Service (GCS, colloquially called 'CE')[1]: the set of software and systems that form the interface between the Grid and a site's local computing facilities and computing elements (representations of the batch queues).

However, a GCS can be implemented in different ways that do not necessarily allow for the mapping to be completed at the time of initial authentication, either because the authenticating process is not capable of setting the Unix identity for the client (being run under a non-superuser account), or because the identity used to submit the actual workload is not yet known at submission time ('late binding'). In these cases, an external utility is needed to perform this mapping, needing super-user privileges to change the uid and gid of the process: a utility such as gLExec. In this paper we have identified three scenarios explicitly:

Non-privileged context acceptance (site-CE) The process that accepts and authenticates the connection does not have sufficient privileges to change Unix identity, e.g., because it runs in a web services container environment with a uid larger than zero. All services that accept and authenticate connections are under site control. The job flow in this scenario is shown in Fig. 1a.

Site-local VO scheduler (VO-CE) The process that accepts the connections is not under site control, but is non-site-managed code that runs as an unprivileged user. This 'site-local scheduler' accepts incoming job requests and subsequently submits jobs to the site's local compute environment, as shown in Fig. 1b.

Traditionally, such a site-local scheduler or job monitor has been run with the credentials of the original user, as in the case of the *gridmanager* in Condor-G [20], but in larger deployments this leads to an unacceptable load on the system running the site-local schedulers, as it incurs a load of at least one running process per user per workload source.

[1] such a service is colloquially referred to as a 'CE', meaning in this sense a physical or virtual machine on which such a system is running. Unfortunately this clashes with the GLUE [6] definition of Compute Element (CE) and in the current context would cause a relatively large amount of consternation. Throughout this paper CE is also used in its colloquial meaning.

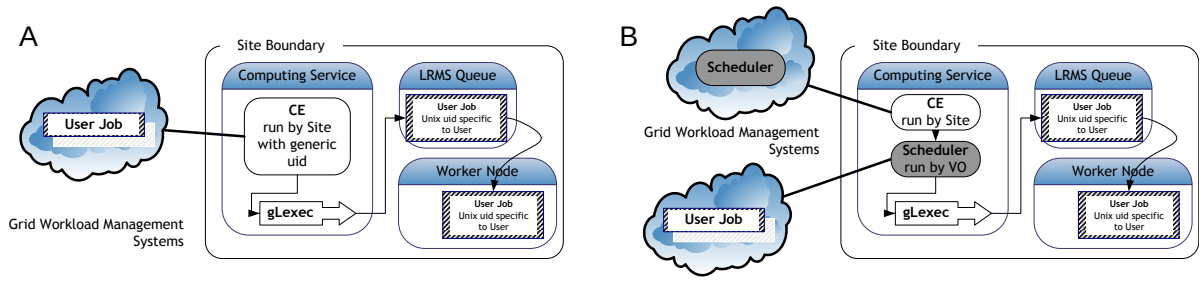

Figure 1. Flow of a grid job through the Grid Computing Service and the LRMS for the Site-CE scenario (A) and the VO-CE scenario (B). White boxes represent processes run under control of the site, gray boxes are processes run by a (generic) VO uid on the Grid Computing Service system, and dashed boxes represent processes or Jobs run or held with a Unix uid associated uniquely to the User owning the workload of this Job.

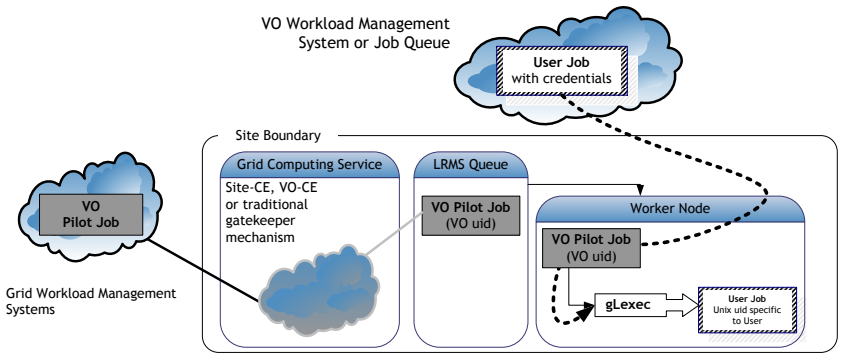

Figure 2. Flow of a grid job through the Grid Computing Service and the LRMS for the pilot job scenario. The Pilot Job can be submitted to the LRMS in various ways that are not relevant to the scenario. Shading as in Fig. 1.

The ability to change Unix identity allows the use of community schedulers that can serve multiple users within the same local scheduler process, whilst retaining the characteristics of per-user job submission by changing Unix identity before any workload is presented to the site batch system.

Here, it is assumed that a trust relationship exists between the individuals or organization running the community scheduler on the site, and the site's management.

Late binding (gLExec-on-WN) Late binding refers to the submission model where a generic 'pilot job' retrieves the actual workload only once resources are allocated to it, *i.e.*, at the time the pilot job has started execution on a computer system – using a job slot on a worker node (WN), in case of systems managed by a local resource management system (LRMS) – as shown in Fig 2.

A pilot job (placeholder job) is a compute job that will not in itself do any work, but instead will retrieve the actual workload from a pre-determined location provided to it at submission time. The submission of pilot jobs can be done either by the user, or by a third party on behalf of a user or group of users. Only in the latter case will an identity switch likely be considered necessary to assert the actual workload's owner to the site and provide Unix-level insulation between workloads from different users on a multi-user system.

In the gLExec-on-WN case, its use is cooperative: it relies on the pilot job on complying with the invocation policy, and in its respecting the result of the authorization decision made in gLExec.

In each of these cases, an external utility is needed, but the way in which that utility is deployed (with or without super-user privileges) is open. There are four possible deployment models, whose choice is largely orthogonal to the scheduling scenario:

Identity-mapping (IM) model To exploit the account mapping capabilities of gLExec, the executable must be run with super-user privileges. This can be accomplished by having the gLExec binary owned by the super-user and enabling the *setuid* bit in the file's mode on a set-uid capable file system. The local credential change on execution of the client program enables isolation between processes run – and files accessed – by different grid identities, and allows distinguishing between the invoking process and the clients started by it.

Non-privileged (NP) model The gLExec binary can be installed without any special privileges. In this case, the authorization and logging facilities of gLExec can still be used, but the actual credential mapping enforcement stage is omitted. In this mode, inter-process and file isolation is not possible. Also, this mode is not suitable for the site-CE and VO-CE scenarios, since the client program will run with the same uid and gid as the GCS and could thus manipulate it, and this manipulation would extend beyond the VO or user that - in case of a pilot job submission - knowlingly induced the possibility for such manipulation.

Site-isolation (SI) model The gLExec binary is made *setuid*, but to a dedicated 'execution account' instead of the super-user, and only the authentication and authorization modules are retained in the configuration (enforcement modules to apply the credential change usually require super-user privileges that are not available in this case). In this mode, inter-job isolation is not possible, but by virtue of the *setuid* local credential change the client will not be able to tamper with the starting process. This mode is appropriate for *canned job submission*, where only known programs are executed and inter-job isolation is less relevant.

No-Op model In case only simple invocation-compatibility is required without the need for authorization or credential mapping, a *no-op* wrapper that executes the specified client directly can replace gLExec (the 'no-op model').

It is clear that to attain proper job isolation and tracability only the "Identity Mapping" mode is appropriate. However, there may be operational or policy reasons why having a program that is *setuid* to root on worker nodes is considered inappropriate. When gLExec is used as part of the site-CE or VO-CE scenario, its invocation bing entirely site controlled and its powers limited to a specific (set of) dedicated nodes, such operational and policy considerations are usually weighed differently.

2.1. Implicit versus Specified Local Credential Determination

The local credential to be used for the execution of a particular job can either be supplied as part of the job description, or can be determined implicitly from the grid identity used to submit it. Although the former provides more flexibility, the latter is the most prevalent in current implementations. Only recently have job submission languages allowed the expressed on a local user name preference (in particular JSDL [21], WS-GRAM RSL [8]) with multi-user Grid Computing Services.

All current gLExec versions, and the LCAS and LCMAPS frameworks described below, use implicit credential mapping to offset the lack of target-user information in the job submission. The local credential in this mode is derived from the identity and attributes used to establish the job submission security context, and may additionally include mapping obligations specified through any external LCMAPS modules.

If explicit credential mapping is desired, the LCMAPS framework, and thereby gLExec, can collaborate with the Work Space Service (WSS) [9] to allow the use of designated 'execution

environments', by virtue of sharing a common back-end interface in LCMAPS. This scenario should be considered if gLExec is used as part of a site's Grid Computing Service.

3. Requirements

As the interface between the grid world and the local site, gLExec should address the grid security requirements [22], both in helping the general security architecture to meet these requirements, and in its own design and implementation.

3.1. Access control in gLExec

In gLExec there are three control points where limitations can be applied:

invoker identity – not all processes or users are necessarily allowed to change Unix identity. To prevent abuse and inadvertent disclosure of site policies, gLExec invocation can be limited to specific uids. Further restrictions can be defined when central authorization services are used based on the credential used to authenticate to the service (either the host or a VO or user credential), or on the source's network address.

target Unix identity – it is generally desirable to limit the allowed range of Unix identities to a subset of the uid space. In particular, limitations are usually imposed such that mappings to privileged (uid 0) or system (uids below[2] 100) accounts are not allowed.

target Grid identity – this limitation implements the grid authorization system as conventionally used in site access control, i.e., authorization decisions as commonly used on the CE today. These policy decisions are based on the grid identity of the User and any attributes carried by the user including their VO-asserted attributes, in general cryptographically protected assertions in the form of Attribute Certificates [23] bound to the grid identity of the user, that are issued and signed by the Attribute Authority of the VO. These are then carried along as an extension in the proxy certificate [3] delegated by the user, such as in the VOMS [4] system.

Target Grid identity authorization in gLExec is identical to the site access control functionality required in the *gatekeeper* scenarios. gLExec thus shares common site access control and credential mapping frameworks used for other grid components (gatekeeper, GridFTP server, and the GSI-enabled ssh server) in the gLite software stack: LCAS and LCMAPS, described later in this paper.

3.2. Centralized versus Machine-local Configuration

In large deployments, it may be advantageous to use a single, centralized point where all policy and credential mapping state is kept. This is especially useful in case multiple GCS systems are used, or when the compute and Grid storage services share a common set of POSIX-style file systems that rely on uid and gid access controls, e.g. the Network File System NFS. Similarly, this mapping consistency is relevant in cluster computing scenarios, where consistency in uids and gids on the worker nodes and on the batch server is required, or where processes on worker nodes may communicate amongst themselves based on uid/gid authentication (such as in many implementations of MPI [24]).

In the gLExec-on-WN scenario a machine-local mapping may be more appropriate, since the number of pool accounts that needs to be available at any one time can be equal to the number of Job Slots available on that specific machine. This obviates the need for the large number of pool accounts that would be needed in case the mapping would span an entire cluster or Unix

[2] The conventions used to designate system accounts is dependent on the Unix Operating System flavour and distribution

system management domain. Once a job terminates and the Job Slot is released, the associated pool account can be re-used for the next job.

Either model must be implementable, and to this end the policy, any persistent state used in mapping, and the enforcement should be distributable over more than one system. In the framework discussed below, this can be achieved by distributing the various functionality modules of *LCAS* and *LCMAPS* over machine-local, cluster-local, or site-wide services.

3.3. Persistency of credential mapping

Any pool-account or pool-group credential association established by gLExec through LCMAPS is semi-persistent: the uid or gid will not be automatically released. This persistency for the duration of the job is required since the mapping engine may need to be invoked repeatedly to steer the local process execution from the grid level (e.g. forced renewal of proxy credentials, management of input and output *sandboxes*) and job management at the local site. Consistency of mapping between multiple invocation of the mapping engines and gLExec on any specific system must therefore be ensured.

However, longer-term persistency that allows re-use of a user's data and software stored in local POSIX-style file systems (long-term persistency) can have important side effects. The ability to store data encourages the submission of jobs leveraging the implicit assumption that data is preserved between job executions on a specific site. For this reason, long-term persistency is uncommon in larger-scale production environments, and local account state clean-out processes are employed to dissociate the mapping and remove any persistent state related to the account at the site.

4. Implementation

gLExec is a stand-alone executable that can validate incoming grid credentials, make an authorization decision based on the credentials and the program to be executed, establish a mapping between the grid identity and a local Unix credential mapping, and subsequently execute the specified binary program with these Unix credentials.

gLExec uses the authorization and mapping systems of the gLite site access control system: the Local Centre Authorization Service (LCAS) and the Local Credential Mapping Service (LCMAPS). Authorization and credential mapping steps are separated, since the binary authorization decision is stateless and can potentially be computed faster than the mapping to a site-local credential. Secondly, it is important that a negative authorization decision does not change the internal state of the site – whereas a state change is a likely in case of credential mapping, e.g. when assigning dynamic or pool accounts. The LCAS and LCMAPS systems are described in mode detail elsewhere [26].

The gLExec code is derived from the `gsexec`, which in turn is derivative of the *suexec* [12] sources, and has been modified to interface to the LCAS authorization service and the LCMAPS credential mapping service. Contrary to *suexec*, gLExec also uses a run-time configuration file, to be made writeable for the super-user only. This protected configuration file contains selected security settings of gLExec. This is needed for a realistic large-scale deployment scenario in a grid where sites are autonomous but are not able or willing to re-build code from source.

The security features of *suexec* that are compatible with the grid use cases discussed in this paper have been retained, whereas checks solely required for the CGI use case have been removed. New control points have been introduced where needed.

no hard constraints on the path of the command executed – *suexec* only allows the execution of commands in the current directory, i.e., leading forward slashes and relative paths containing the parent directory are prohibited in *suexec*. This restriction has been lifted in gLExec, but the LCAS RSL access control plug-in provides an alternative extended mechanism to restrict the execution of commands.

no closure of open file descriptors – gLExec will retain open file descriptors across the invocation and uid change, and the target user job will avail over any open files, including the standard input, output and error streams and pipes of the parent process. This semantics is used by in the Site-CE scenario (e.g. by the CREAM CE [28]), and by the Condor-G [20] glide-in system. In the gLExec-on-WN scenario, pilot jobs should close any inappropriate files before invocation.

checks on user and group id of those allowed to invoke gLExec – in *suexec* only the (pre-defined) owner of the httpd process is allowed to invoke *suexec*. gLExec replaces this with a (super-user only) configurable white list of users allowed to invoke gLExec, to allow specific processes (in the Site-CE scenario) or a range of accounts (in the gLExec-on-WN scenario) to invoke gLExec.

changes to environment cleaning – in *suexec* the environment is reset to include only a single pre-defined path. gLExec can preserve a configurable set of environment variables (defined in the protected configuration file), but will always unset the `LD_*` variables and will re-set `PATH` to a value defined at compile-time. The HOME variable will be re-set to the one of the target uid.

As part of the execution, a delegated proxy credential may be copied to the filesystem in the target user's uid space. The use of a copy instead of a delegation of the proxy is appropriate in this case, as the credential does not leave the machine across a gLExec invocation (the source of proxy should of course be appropriately delegated). File type and ownership requirements are checked for both source and target location to ensure this mechanism cannot be abused.

Other than for the user proxy, gLExec does not provide a mechanism to transfer files between the invoker of gLExec and the target process. The retention of open file descriptors across the gLExec invocation, and the guaranteed mapping consistency between repeated invocations of LCMAPS with the same grid identity, provide the elements on which such a facility can be built.

In order to allow logging of the child's process execution times, a `fork-execv` construction can be used in place of the simple `execv` used in the original *suexec* code. gLExec will then `waitpid(2)` on the child and log the the return code of the child, and the real, user, and system time spent by the child, to `syslog`. This behaviour is configurable via the protected configuration file.

4.1. gLExec semantics with respect to the operating system environment

When used in the gLExec-on-WN scenario, it is important that the process tree and session affinity is not broken by the use of gLExec, and that gLExec appropriately accumulates times for its child processes. Batch systems rely on this process tree to track and trace processes that are executing on a particular node, and assume that resource usage information about all processes started as part of one job are accumulated by the parent process (using `wait` or `waitpid` [27]). However, gLExec cannot ensure that the user processes started by it appropriately wait on any processes they start subsequently.

5. Policy considerations

In general, the policy considerations to deploy gLExec in the Site-CE scenario is comparable to those faced when deploying a network gatekeeper with super-user privileges. Given this similarity, such considerations are not discussed in this paper.

In the late binding scenario where the pilot jobs are submitted by a third party ('the VO') and can bind to workloads submitted by a variety of users, novel security policy issues may arise. E.g., at the site level it is not obvious who is responsible for the workload running on a particular WN at any one time. The knowledge on workload ownership is distributed between the site (who sees the owner of the pilot job), state kept within the pilot job (the workload it

is bound to), and information kept by the third party responsible for submitting the pilot job and its instructions (the third-party's management framework, usually a VO). Depending on site-local circumstances, this distribution of knowledge may or may not be acceptable,

The use of gLExec in the VO-CE scenario, and by the pilot job for gLExec-on-WN, involves declaring the grid identity of the target user directly to the site, and aims to address this issue. In either case, the use of gLExec is cooperative as it assumes the responsible VO-CE or pilot job will truthfully call gLExec and respect gLExec's decision to run or refuse a particular target user job. This cooperation cannot be a-priori enforced by technical means, and relies on appropriate policy agreements between the site and the VO or user. Failure to comply with the policy can be established a posteriori by inspection of accounting logs maintained the batch system (VO-CE scenario) or the process accounting logs on the worker nodes (gLExec-on-WN scenario).

Without local credential mapping, as performed by gLExec, it is hard to impossible to distinguish at the process and system call level between the pilot job and the executables and script run by the pilot job on behalf of a third user. Even if the start of a user job is declared by the pilot job to the system via a logging mechanism, individual processes that are not part of the processes tree of the pilot job or otherwise 'escape' will be using the same local uid and gid as the pilot job. Since multiple pilot jobs may be executing on the same system (in case a single system offers multiple independent job slots), it cannot be determined which user is responsible for any behaviour observed on the system, files will be shared, and the target user has access to the pilot job data, including any proxy credentials of the pilot job. By using gLExec in Identity Mapping mode, each target user is contained in its own uid and gid domain, and standard operating-system mechanisms can be used for auditing, accounting and logging of such processes. In combination with audit capabilities available in systems that meet Evaluation Assurance Level 3 of ISO/IEC 15408 [29], this can provide enough information to associate any individual action on a WN to a specific grid identity.

In the gLExec-on-WN scenario, using gLExec in identity mapping mode does entail running a super-user *setuid* binary. Such a deployment obviously poses its own security risks and associated policy isses. The tradeoff between the different risks and responsibilities ultimately remains a site-local decision.

6. Deployment

The LCAS and LCMAPS subsystems, responsible for most of the authorization and mapping functionality in gLExec, were introduced in the middleware deployed by the EU DataGrid [30] project in its 2.0 release in 2003, in conjunction with a modified version of the Globus gatekeeper (*edg-gatekeeper*) and GSI-enabled wu-ftpd server (*edg-gridftpd*) [17]. The code base was subsequently forked for use both in the middleware deployed by the LHC Computing Grid Project [31] in 2004 with frozen functionality, and in the gLite [25] software stack of the EGEE [32] project where it has been further developed. It has been included as a standard component by the aforementioned deployment projects in order to support fine grained authorization and resource management based on VOMS attributes.

gLExec is available as a released component in gLite version 3.1 and higher. A pre-release version of gLExec has been deployed on production-level resources at the Fermi National Accelerator Laboratory (FNAL) in the gLExec-on-WN scenario [33]. In conjunction with the GUMS system it is used to enable explicit job separation and identification of the user responsible for any specific job on a worker node. Since the initial deployment of a gLExec version in October 2006, no major obstacles were encountered.

Following the deployment of gLExec at FNAL it is current (August 2007) being introduced as a component in the Virtual Data Toolkit [34] for use on the Open Science Grid [35].

gLExec is also used as part of the CREAM CE system for a Site-CE scenario. The CREAM CE is based on web services and implemented in Java. The CE system itself therefore runs in

an unprivileged JVM container, and uses gLExec to switch uid in the Unix domain before it submits jobs to the LRMS.

7. Related work

Mapping between grid identities and local Unix credentials has been part of all computational grid services, usually handled as part of the service accepting the incoming request.

The Globus [1] pre-web services gatekeeper provides an integrated authorization and mapping system based on a many-to-one *grid-mapfile*, where one or more X.509 subject names were associated with a single Unix uid. The credential mapping is an integral part of establishing the network connection and the SSL/TLS handshake, and is combined inside a single daemon process running with elevated privileges.

The *grid-mapfile* many-to-one mapping was extended by McNab via the `gridmapdir` mechanism [7]. Pool accounts provide a unique one-to-one mapping without the need to pre-assign unique uids to each grid identity: it is assigned only once a mapping is actually requested, and is then persistent until cleared explicitly. The format of the *grid-mapfile* used for the gridmapdir system is backwards-compatible with the one used by Globus. The pool account system is used in the mapping framework on which gLExec is based.

The WS-GRAM service [8] that is part of Globus Toolkit version 4 does not change local identity as part of the network daemon, running entirely inside a Java virtual machine (JVM). The WS-GRAM service performs a lookup of the local Unix identity (many-to-one using a grid mapfile, or an explicit many-to-many mapping using the Work Space Service [9]) whilst running inside the JVM. It then uses *sudo* [10] to change identity from the generic userid used to run the JVM to the designated local identity. It is limited to a fixed set of accounts per grid subject name, listed in the grid-mapfile, whereas gLExec and the LCMAPS framework instead provide an implicit mapping, based on the grid credentials provided.

A change of credentials is also employed in the *suexec* ancillary program to Apache *httpd* [11, 12]. When used with *httpd*, *suexec* allows running CGI [13] programs as the owner of the CGI program, instead of the user id under which the *httpd* program is run.

For integration with the *GridSite* system [14], a like program *gsexec* was developed. Derived from Apache's *suexec* code base, it enables the use of grid credentials to determine local Unix identity to programs launched via Apache's *httpd* and the *mod_gridsite* module, including the use of pool accounts. It is designed to act as a drop-in replacement for *suexec*. gLExec in turn is derived from *gsexec* and augments its functionality by providing the pluggable authorization and credential mapping frameworks LCAS and LCMAPS, and by allowing for the preservation of file descriptors across the gLExec invocation. This changes the invocation semantics of gLExec with respect to *suexec* and *gsexec*, but allows a wider range of use cases that do not involve the use of a web server.

The *sudo* program [10] allows permitted users to execute commands as the super-user or, in general, a different user on the local system. Execution privileges are granted based on the local userid of the current user and, optionally, on knowledge of both the local user's password and the specific executable that will be invoked. Conceptually providing functionality similar to gLExec, it only uses local, not grid identities for access control.

8. Future Directions

It has been recognised that additional functionality is beneficial in an LCAS and LCMAPS deployment scenario where the authorization decision is made is a highly distributed way, i.e., by using gLExec on many WNs inside a site instead of on a relatively low number of Grid Computing Service systems.

Since almost all of the grid functionality of gLExec is implemented via the LCAS and LCMAPS systems, future work includes development of site-central LCAS and LCMAPS

'services', and authorization modules (LCAS) and acquisition modules (LCMAPS) to communicate therewith. The PRIMA module developed to communicate with the GUMS [18] system is a conceptual example of the desired behaviour. However, since GUMS relies on a site-local copy of all membership data of all VOs supported at a particular site (i.e. the VO attributes are passed along as strings and not as VOMS attribute certificates in the communication protocol), whereas the VOMS attribute validation model used in gLite is based around validation of the incoming signed attribute certificates, an alternative is needed that matches the latter paradigm. Since similar information is exchanged in both the GUMS and the gLite case, a common communications protocol is being pursued that will allow either system to be used interchangeably. Current work, in collaboration with the developers of the Globus Toolkit, the VO Privilege Project and the gLite team, aims to converge on a common protocol and library based on the SAML-XACML binding. Once the VO attributes passed to the site central service remain cryptographically protected, the gLite and GUMS validation models converge and also GUMS could rely on the incoming assertions and thus would no longer need to cache all VO information locally.

With the availability of an authorization communications protocol for LCAS and LCMAPS, also cross-site central ban lists can be supported, allowing improved emergency response to compromised grid identities and VO attributes. It remains to be seen how far the protocol can be used for communications with a generalized credential validation service.

It should however be noted that also in the absence of a site-central service for policy configuration, all gLExec deployment scenarios, including the gLExec-on-WN scenario, can be realized in a consistent and secure way.

9. Summary and Conclusions

To integrate resources that use Unix-like semantics for user management and access control in a grid environment, site-local authorization and credential mapping are required. The gLExec system, used in combination with the LCAS site-local authorization system and the LCMAPS local credential mapping service, provides an integrated solution for site access control to grid resources. With the introduction of gLExec, the submission model can be extended from the traditional gatekeeper models, where authorization and credential mapping only take place at the site's 'edge'.

Retaining consistency in access control, gLExec allows a larger variety of job submission and management scenarios that include per-VO schedulers on the site and the late binding of workload to job slots in a scenario where gLExec in invoked by pilot jobs on the worker node.

Future extensions of the LCAS and LCMAPS system to use central authorization and mapping services over a standard protocol will further improve deployability and enable better control over the site access control policies in effect.

Acknowledgements

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. Full information is available at http://www.eu-egee.org.

This work is part of the research programme of the Foundation for Fundamental Research on Matter (FOM), which is financially supported by the Netherlands Organisation for Scientific Research (NWO).

References

[1] Foster I, Kesselman C and Tuecke S *The Anatomy of the Grid: Enabling Scalable Virtual Organizations* International J. Supercomputer Applications, **15**(3), 2001

[2] ISO/IEC JTC 1 and ITU-T *Information Technology – Open Systems Interconnection – The Directory: Authentication Framework* ISO/IEC International Standard 9594-8, ITU-T Recommendation X.509
[3] Tuecke S, Welch V, Engert D, Pearlman L and Thompson M *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile* RFC 3820
[4] Alfieri R, Cecchini R, Ciaschini V, dell'Agnello L, Frohner A, Lorenty K, Spataro F, *From gridmap-file to VOMS: managing authorization in a Grid environment* Future Generation Computer Systems **21 (4)** 549–558 (2005)
[5] Ritchie D M and Thompson K *The Unix Time-sharing System* C. ACM **17(7)** 365-370
[6] Andreozzi S, et al. 2007 *GLUE Schema Specification Version 1.3* GLUE Working Group, 2007
[7] McNab A *Grid-based access control for Unix environments, Filesystems and Web Sites* Proceedings of Computing in High Energy and Nuclear Physics, La Jolla, California, USA, March 2003 (TUBT008, ePrint cs.DC/0306030)
[8] Feller M, Foster I and Martin S *GT4 GRAM: A Functionality and Performance Study* submitted to TeraGrid 2007 Conference, Madison, Wisconsin, USA (http://www.globus.org/alliance/publications/papers/TG07-GRAM-comparison-final.pdf, accessed August 22, 2007)
[9] Keahey K, Foster I, Freeman T, Zhang X and Galron D *Virtual Workspaces in the Grid* Proceedings of Europar 2005, Lisbon, Portugal, September, 2005
[10] Miller T C *Sudo Main Page* http://www.gratisoft.us/sudo/ (accessed August 21, 2007)
[11] *The Apache HTTP Server Project* http://httpd.apache.org/ (accessed August 22, 2007)
[12] *suEXEC Support* http://httpd.apache.org/docs/trunk/suexec.html (accessed August 22, 2007)
[13] *CGI - Common Gateway Interface* http://www.w3.org/CGI/ (accessed August 22, 2007)
[14] McNab A *The GridSite Web/Grid security system* Software: Practice and Experience **35(9)** 827-834
[15] Foster I, Kesselman C, Tsudik G and Tuecke S *A Security Architecture for Computational Grids* Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, pp. 83-92
[16] Alfieri R, et al. *Managing Dynamic User Communities in a Grid of Autonomous Resources* in Proceedings of the Computing in High Energy and Nuclear Physics conference, 24-28 March 2003, La Jolla, California, USA (TUBT005, ePrint cs.DC/0306004)
[17] Röblitz T, et al. *Autonomic Management of Large Clusters and Their Integration into the Grid* Journal of Grid Computing **2** 247260 (2004)
[18] The VO Privilege Project *Grid User Management System* http://computing.fnal.gov/docs/products/voprivilege/ (accessed August 21, 2007)
[19] Lorch M *Transition to Role-based Assignment of Local User-Ids* http://computing.fnal.gov/docs/products/voprivilege/documents/transition-to-privilege.html, November 2005 (accessed August 21, 2007)
[20] Frey J, Tannenbaum T, Foster I, Livny M and Tuecke S *Condor-G: A Computation Management Agent for Multi-Institutional Grids* Journal of Cluster Computing volume **5** 237-246 (2002)
[21] Anjomshoaa A, Brisard F, Drescher M, Fellows D, Ly A, McGough S, Pulsipher D, and Savva A *Job Submission Description Language (JSDL) Specification v1.0* GFD.56
[22] Venekamp G *EGEE JRA3 Activity user requirements* EGEE-JRA3-TEC-485295 (version 1.0) (https://edms.cern.ch/document/485295) (accessed August 29, 2007)
[23] Farrell S and Housley R *An Internet Attribute Certificate Profile for Authorization* RFC 3281
[24] Dongarra J J, Otto S W, Snir M and Walker D *A message passing standard for MPP and workstations* Commun. ACM **39 (7)**, 84-90
[25] *Lightweight Middleware for Grid Computing* http://www.glite.org/ (accessed August 22, 2007)
[26] Groep D, Koeroo O, Venekamp G *Grid Site Access Control and Credential Mapping to the Unix domain* Nikhef PDP Technical Report (http://www.nikhef.nl/grid/lcaslcmaps/) (to appear)
[27] The Open Group *Single Unix Specification Version 2.0* Unix98, 1998
[28] Andreetto P, et al. *CREAM: A simple, Grid-accessible, Job Management System for local Computational Resources* in Proceedings of the Computing in High Energy and Nuclear Physics conference, Mumbay, India, February 2006 (to appear)
[29] The Common Criteria Recognition Arrangement members *Common Criteria for Information Technology Security Evaluation* CCMB-2006-09-001 and ISO/IEC 15408 (http://www.commoncriteriaportal.org/, accessed August 22, 2007)
[30] *EU DataGrid project* http://www.edg.org/ (accessed August 22, 2007)
[31] *LHC Computing Grid project* http://www.cern.ch/lcg/ (accessed August 22, 2007)
[32] *Enabling Grids for E-Science project* http://www.eu-egee.org/ (accessed August 22, 2007)
[33] Yocum D, Sfiligoi I and Petravick D *Addressing the Pilot Security Problem With gLExec* Proceedings of the Computing in High Energy and Nuclear Physics conference, Victoria, Canada, September 2007 (to appear)
[34] *Virtual Data Toolkit* http://vdt.cs.wisc.edu/ (accessed August 22, 2007)
[35] *Open Science Grid* http://www.opensciencegrid.org/ (accessed August 22, 2007)