# User Management Issues In PDMS

Abhishek Gaurav, Nayden Markatchev, Philip Rizk and Rob Simmonds
Grid Research Centre, University of Calgary.
*http://grid.ucalgary.ca*

## 1 Introduction

This document discusses issues related to managing user accounts for use with the Proactive Data Management System (PDMS). PDMS has a number of components, all of which need to support some type of authentication and authorization mechanism. The authorization could take place at the user level, where an individual user has to authenticate with a service or resource they are allowed to use, or at the system level, when services communicate communicate directly with each other. In each case the type of authorization used will be indicated and its implementation described.

The rest of this document is organized as follows. Section 2 provides a primer on authentication and authorization mechanisms in Grid environments. Section 3 provides a brief overview of the components of the PDMS system and their authentication and authorization mechanisms. Sections 4 and 5 describe the authorization and authentication mechanisms used to restrict the changes to file meta-data and files respectively. User Account requirements are described in Section 6. Section 7 provides a summary of the document.

## 2 Grid Authorization and Authentication

Grid environments provide a platform that allows a user to dynamically integrate a diverse set of computational, network, storage and data resources from different administrative domains. In order to dynamically integrate resources, a process must have the ability to authenticate with all resources a user has access to without intervention from the user. This single-sign-on capability is essential to a grid environment.

### 2.1 Grid Security Infrastructure

The Grid Security Infrastructure (GSI) [8] is the security component supported in the Globus Toolkit [7]. It uses a certificate authority (CA) model where users own private keys and matching public keys that they have signed by a CA. The CA is responsible for determining that the user requesting a certificate is who they claim to be and for revoking certificates that have been compromised.

Users don't use their private key and certificate directly to access a GSI enabled resource. They generate a proxy certificate, which is a hash of their private key and signed certificate. These proxy certificates have a short lifetime, to limit the amount of damage that could be done if they are stolen. The proxy certificates are used to authenticate with resources. This provides single-sign-on functionality.

Each certificate (and proxy certificate) contains a globally unique Distinguished Name (DN) that identifies the subject. The subject may identify a person or another service. On most resources, this DN is then mapped to a local identity in accordance with local policies.

### 2.2 Virtual Organization Management

A concept that is often talked about in the grid computing community is that of a Virtual Organization (VO). Use of a VO allows group of users to be authorized to use a system by authorizing the VO containing this group of users to access the resource. The membership of the VO is managed separately by the VO's administer. This separates the authorization of a well defined group of users (the ones in the VO) to access a resource, from the management of who is in the particular VO.

The group of people in a VO could be the people in a research group, a university department, a computing consortium or a group working on a collaborative project, possibly from many different locations around the world. It is essential that resource owners trust the way the VO operates, which could initially limit the number of useful VOs in existence. The provision of secure VO management providers could change this.

Currently the most used VO management tool is VOMS [4]. At the time of writing VOMS relies on an old version of the Globus toolkit, so some work will be required to get this working with PDMS which uses the latest Globus Toolkit release. VOMS is also very difficult to build outside the environment it is distributed with which only supports Linux.

VO management tools determine authorization to a resource such as PDMS. Internally PDMS used Distinguished Names directly and wrapping PDMS with an authentication scheme that meets users needs will not affect PDMS functionality at all.

## 3 PDMS Components

PDMS builds on a number of different services all of which have authorization mechanisms. PDMS and the services it uses all communicate with database systems. A description of these can be found in [11]. It also stores data on and replicates data between file-systems. Most of these file-systems are on Unix servers, though some may be on other operating systems.

For basic PDMS operations, access to the file-systems that store the data being managed is performed using GridFTP [5]. This allows authorized users to read and write data files to file-systems. GridFTP uses the GSI authentication scheme described in section 2.1. The ability for a user to read or write to a particular file or file-system depends on the local file-access attributes. This issue is discussed in Section 5.

There are some tasks that would be difficult to do without PDMS having access to a shell on servers with access to the file-systems. One such task would be to keep a checksum for each file. The checksum itself could be kept at another location, but the ability to compute the checksum and validate the file does require shell access. GridFTP does support shell access to commands in its own restricted binary directory, but to support the checksum operation site owners would need to install the checksum program in this directory. They may not wish to do this since computing checksums would add load to their system. Alternatively, as long as the system has somewhere to store checkpoint files, the checksum could be validated at the point that a file is checked out of the PDMS system.

Other services used by PDMS include the replica location service (RLS) [6], the metadata catalog service (MCS) [13] and the reliable file transfer service (RFT) [9]. RLS is a highly scalable distributed database that maintains the physical locations of logical files. MCS is a database service that designed to maintain arbitrary metadata about logical files. RFT provides fault tolerant file transfer services. Issues with accessing RLS and MCS are discussed in section 4. PDMS uses proxies delegated by the client when making a request to RFT and no special requirements arise from the use of it.

Access to the main PDMS server is controlled using GSI and a mapfile that maps authorized DNs to local user accounts. The mapfile used is a configurable option, so the mapfile used by other services can be shared by PDMS, or PDMS can have its own grid mapfile.

The PDMS server may have its own server certificate, or may share one with other services on the machine, such as GridFTP or GSISSH. It is usually preferable to give the PDMS service its own certificate because there is no need for PDMS to be run as root. PDMS uses this certificate for authentication with the clients.

# 4 MetaData Restrictions

Both RLS and MCS store metadata. MCS stores information that maps the description of a file to a logical file name (LFN) used as a key in RLS. RLS stores mappings from the LFNs, to Physical File Names (PFNs) of actual instances of the files described. One of the gaps in current file catalogue services such as MCS and RLS is the varying granularity of security access. MCS allows control of access to individual logical files files or collections of logical files while RLS only allows access to be defined on a per RLS server basis. One of the functions of PDMS is to provide flexible access restrictions that meet the needs of any group of users.

The MetaData access restrictions on PDMS are primarily at the level of collection and are defined by a series of Access Control Lists (ACLs) for a collection. A user may be granted certain levels of access to a particular collection by having the distinguished name on their certificate added one of the following ACLs:

- *Read*: The user has the ability to query the meta-catalogue of have replicas of data pushed to locations.

- *Write*: The user has the ability to register and delete metadata or the physical data files. they have the ability to modify metadata.
- *Manage*: The user has the ability to modify the *read* and *write* ACLs.

The computational cost of implementing this access control is small. It is possible to disable this security function to slightly increase performance. Although it is possible in the future that PDMS could define ACLs for each individual logical file entry (rather than just collections), use of such a feature would severely degrade performance. Checking the ACL for every individual logical file operation would slow down performance considerably

## 4.1 RLS Control

PDMS uses RLS to register the physical locations of logical files to inter-operate with various workflow tools such as those provided with the virtual data toolkit (VDT) [3]. VDT is a toolkit designed to develop and execute abstract work-flows. By using the preexisting metadata services such as MCS and RLS, the data stored with PDMS will be accessible to the other tools without modification. This is what is meant by interoperability in this document. PDMS is also required to provide flexible access control to both data and metadata. This is referred to as the security requirement. This section discusses the implications of these requirements. RLS does not allow access to be restricted on a per collection or logical file basis. This makes it difficult to meet all interoperability and security requirements that may be desired by some groups. There are several options, each with different benefits and drawbacks:

- *Restrict read and write access to RLS to PDMS.* This option is relatively easy from a development perspective and is the most flexible with respect to meeting a user groups security demands. The cost of this option is that requires that PDMS respond to all logical file location requests on behalf of users. This will entail a severe performance penalty as RLS uses its own protocol and is designed for quick response, while PDMS is web service based for interoperability reasons. It also does not meet the interoperability requirement.
- *Restrict write access to RLS to PDMS, allow read access to RLS by all.* This option meets the interoperability requirement and is the easiest to develop. It also meets the security requirements of a large number of groups, who are not really concerned about outside entities reading this information, so long as they cannot modify it. It is conceivable that some user groups may want to be able to modify RLS with other tools, but this is a small set of users as PDMS is designed to manage replicas for groupsW. For this reason, this is the option that is currently implemented in PDMS. The bigger concern is some groups may not want the metadata stored by RLS to be available to the world.

- *Use a separate RLS for each VO, and allow PDMS to manage access to it.* If an RLS is used for each VO, it would be possible to meet the security and interoperability requirements. A prototype service was developed by the University of Calgary Grid Research Centre to asses the feasibility of this. It is possible to dynamically deploy multiple RLS services at the same site, using multiple databases on the same database management system. PDMS could conceivably deploy these and manage the authorization configurations for each. This would be manageable because each VO would simply have its own RLS. This solution assumes that a VO does not require permissions to vary by by collection. Using a separate RLS for each VO makes development and deployment more difficult. Although the development costs are not a large concern, heavy deployment costs will discourage adoption of the system.

# 5 Ownership of Files

When planning a PDMS deployment, a decision must be made about who owns the files on the individual file systems. Either the users, or the PDMS system itself must own the files. This section discusses the advantages and disadvantages of each. Currently, PDMS is implemented in a manner that assumes individual users own the files. Modifying this may require significant changes to the semantics of checking files in and out of the system.

## 5.1 PDMS service ownership of files

The advantage of PDMS retaining ownership of the files is it enables the system to provide stronger consistency guarantees. When a file is checked into PDMS, the service can ensure that the file is not removed, as it is the only one that can do this. This allows the system to provide stronger assurance that the files are managed appropriately, and that master copies that exist to ensure data persistence and integrity are maintained.

PDMS ownership of files incurs costs, however. In order to grant a user access to data, a copy of the file would have to be made, even if a replica already exists the system. If a copy is made, this entails extra read, write and storage costs.

The access costs associated with PDMS maintaining ownership of the files does not exist if read only access is granted to the user in some way. If the system granted read access to the users, it would have to do so in a way that only grants access to the appropriate users. The two mechanisms to accomplish this are group access and ACLs.

Setting appropriate group ownership on files will allow read access to any member of the group defined to own the file. This solution requires a registry that provides information about groups. In many real world deployments a registry is not even required. Many large research use a standard name for their group on all machines they have access to, however this does limit the scalability of the system. Several technologies are amenable to functioning as a registry for group and other file permissions information. The simplest solution is to store this information as metadata associated with collection in the MCS database. A more scalable solution may be to store group information in a standard format on a technology

such as Shibboleth [2]. Shibboleth is an opensource middleware project to provide single-signon, but it also functions as a very secure, configurable directory service or registry for users.

Using ACLs is a more robust approach. ACLs allow fine grain file access control for any number of users or groups. Unfortunately, ACLs are not supported on all file systems. In addition to the lack of universal support, although there is a POSIX standard, the standard tools to manipulate ACLs are not identical. The lack universal, uniform support for ACLs makes it difficult to implement the manipulation of them. The information services required to support use of ACLs in an environment with multiple operating systems is not currently available. Models of the systems being used would have to be provided [1]. Currently, discovery services designed to distribute this information exists, but there is no system to generate the information about the systems.

In addition to the above real costs, there exists an opportunity cost with respect to storage facilities. It is likely a user of PDMS would have access to storage facilities that PDMS does not. One of the goals of PDMS is to allow groups to integrate data storage facilities across administrative domains. This is relatively simple when users own the files, because sites would simply have to have a GridFTP server and would not need to be aware of PDMS. This is not likely possible when data is owned by PDMS because it requires accounts to be created for the PDMS service itself. It is not practical to integrate a storage resource a user has access to by giving PDMS an account because PDMS may store files for users that currently do not have access to the system in question. By giving PDMS access to their system, administrators are essentially delegating authorization to use the system to an outside service. Few administrators would be willing to do this.

## 5.2 Individual ownership of files

The advantage of individual ownership of files is flexibility. When individuals maintain ownership of files, and PDMS acts on their behalf using proxies delegated by the user, the system can easily assist them in integrating separate storage facilities. Access to files is also facilitated as users can read replicas directly if they do not intend to modify them.

The main disadvantages of individual ownership is that inconsistencies in the data are more easily introduced and sharing data between members of the same VO is more complicated.

Users can more easily introduce inconsistencies by deleting, moving or modifying registered files. The effects of this can be mitigated by making files registered with PDMS read only, moving them to specific locations where it is clear that PDMS is aware of them, and making PDMS actions robust to the possibility that files have been deleted. Checking of file contents against a recorded hash of the file may also help. This could be done either during spare cycles or explicitly at the request of the user. How to accomplish this most effectively still an open problem. In a related project, an hash of a file was taken using JAVA libraries before checking a file in. This hash was then added to the metadata of the file and was used to validate a file upon checkout. This was found to be effective.

Sharing data between members of a VO become more complicated as a file could be owned

by one member, while other members will not have read access to them. When attempting to grant multiple users access to files, the same problems of using group file access and ACLs as discussed in section 5.1.

# 6 User Account Requirements

This section describes the account requirements on a Unix system for each component of the system. These components include the PDMS server, RLS, MCS, RFT and GridFTP.

The PDMS service itself needs a Unix account to run under; users of PDMS do not. The choice of the account used for the service is quite flexible. If necessary for administrative reasons, the PDMS service can be run by a normal user. This is normally only done when the VO does not have administrative access to create a special user account. The user would need access to a DBMS used by PDMS as well as access to the file system to write logs. The user would in effect have access to the proxies of all users of the PDMS system which is not desirable. It is generally preferable to create a separate user for PDMS. This restricts access to proxies to people who have access to the PDMS user account. It also follows the general security principle of restricting servers to the minimum access required by the service. PDMS does not need root access. Note that the threat of proxies being stolen is always an issue, regardless of software used. If the admin account on a machine is compromised all proxies on that machine have also been compromised. This is the reason proxies have a limited lifespan.

If a virtual organization needs to provide the MCS service because they use workflow tools that depend on it, this will require and account and there is little benefit and increased complications if it a different user than the account used for PDMS. PDMS currently uses RFT for reliable file transfer. This RFT may be run anywhere under any user.

RLS needs at least one account per class of user. A class of user implies each user within a class has the exact same account access and can be mapped to the same UNIX account . Note this is the same user account not group. This is a result of the way RLS defines their ACLs. If PDMS is managing the RLS servers as described in Section 4.1, then PDMS will need an account for each access class it creates. This in turn implies that PDMS needs to have access to a pool of accounts that correspond to these access classes. These accounts do not need passwords or home directories.

In most cases, the GridFTP servers need accounts for each user that is mapped from the DNs of users. This depends on the mapping algorithms used.

# 7 Summary

This document summarized the authentication, authorization and other security mechanisms within the PDMS system. The PDMS system utilizes several other services including RLS, MCS and GridFTP, each with their own security models. Diversity in file systems creates security challenges, because of differing support for ACLs and storage resources may come

from more than one administrative domain. The use of registries and a robust model of system information is required for some usage scenarios. For a full architectural overview of the PDMS system, refer to [12]. For a discussion of database issues within PDMS see [10].

# References


[1] Model document placeholder.

[2] Shibboleth. http://shibboleth.internet2.edu/.

[3] Virtual data toolkit web site. http://vdt.cs.wisc.edu.

[4] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lorentey, and F. Spataro. From gridmap-file to voms: managing authorization in a grid environment. *Future Generation Computer Systems*, 21:549–558, April 2005.

[5] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. GridFTP: protocol extensions to FTP for the Grid. Proposed Recommendation GFD-R-P.020, Global Grid Forum, April 2003.

[6] A. Chervenak. Giggle: A framework for constructing scalable replica location services. In *Proceeding of the IEEE Supercomputing 2002*, 2002.

[7] I. Foster. Globus Toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13, 2005.

[8] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *Proceedings 5th ACM Conference on Computer and Communications Security Conference*, pages 83–92, 1998.

[9] R. Madduri, C. Hood, and W. Allcock. Reliable file transfer in grid environments. In *Proceedings of 27th Annual IEEE Conference on Local Computer Networks*, pages 737–738. IEEE press, 2002.

[10] U. of Calgary Grid Research Centre. Database assessment for pdms. GRC Internal report.

[11] U. of Calgary Grid Research Centre. PDMS database issues. GRC Internal report.

[12] U. of Calgary Grid Research Centre. Proactive Data Management System. GRC Internal report.

[13] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A metadata catalog service for data intensive applications. In *Proceedings of Supercomputing 2003 (SC2003)*, 2003.