

An Analysis of Security Services in Grid Storage Systems

Jesus Luna, Michail Flouris, Manolis Marazakis, Angelos Bilas
{jluna, flouris, maraz, bilas}@ics.forth.gr

*Institute of Computer Science (ICS),
Foundation for Research and Technology - Hellas (FORTH),
PO Box 1385. GR-71110. Heraklion, Greece.*

Federico Stagni
{federico.stagni}@fe.infn.it

*Istituto Nazionale di Fisica Nucleare sez. di Ferrara,
via Saragat 1 - 44100 Ferrara, Italy*

Alberto Forti, Antonia Ghiselli, Luca Mangoni, Riccardo Zappi
{alberto.forti, antonia.ghiselli,
luca.mangoni, riccardo.zappi}@cnafe.infn.it

*Istituto Nazionale di Fisica Nucleare CNAF,
viale Berti Pichat, 6/2 - 40127 Bologna, Italy*



CoreGRID Technical Report
Number TR-0090
August 31, 2007

Institute on Knowledge and Data Management

CoreGRID - Network of Excellence
URL: <http://www.coregrid.net>

An Analysis of Security Services in Grid Storage Systems

Jesus Luna, Michail Flouris, Manolis Marazakis, Angelos Bilas
{jluna, flouris, maraz, bilas}@ics.forth.gr

Institute of Computer Science (ICS),
Foundation for Research and Technology - Hellas (FORTH),
PO Box 1385. GR-71110. Heraklion, Greece.

Federico Stagni
{federico.stagni}@fe.infn.it

Istituto Nazionale di Fisica Nucleare sez. di Ferrara,
via Saragat 1 - 44100 Ferrara, Italy

Alberto Forti, Antonia Ghiselli, Luca Mangoni, Riccardo Zappi
{alberto.forti, antonia.ghiselli,
luca.mangoni, riccardo.zappi}@cnafe.infn.it

Istituto Nazionale di Fisica Nucleare CNAF,
viale Berti Pichat, 6/2 - 40127 Bologna, Italy

CoreGRID TR-0090

August 31, 2007

Abstract

With the wide-spread deployment of Data Grid installations, and rapidly increasing data volumes, storage services are becoming a critical aspect of the Grid infrastructure. Due to the distributed and shared nature of the Grid, security issues related with state of the art data storage services need to be studied thoroughly to identify potential vulnerabilities and attack vectors. In this paper, motivated by a typical use-case for Data Grid storage, we apply an extended framework for analyzing and evaluating its security from the point of view of the data and metadata, taking into consideration the security capabilities provided by both the underlying Grid infrastructure and commonly deployed Grid storage systems. For a comprehensive analysis of the latter, we identify three important elements: the *players* being involved, the underlying *trust assumptions* and the dependencies on specific *security primitives*. This analysis leads to the identification of a set of potential security gaps, risks, and even redundant security features found in a typical Data Grid. These results are now the starting point for our ongoing research on policies and mechanisms able to provide a fair balance between security and performance for Data Grid Storage Services.

1 Introduction

Storage systems have become an essential piece for the data Grid, thus making it imperative to establish an integral and standardized security solution able to avoid common attacks on the data and metadata being managed. Grid

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

security research has focused in specific high-level services (for example GSI [1] and VOMS [2]) instead of providing a systemic view able to encompass even block-level security features. Work-groups like the Open Grid Forum's OGSA Data Architecture [3] and CoreGRID's Trust and Security [4] have begun to investigate the challenges related with providing security at the Grid Storage Level.

In this paper we apply to a typical use case for the Data Grid (encompassing data creation, replication, discovery and retrieval) an extended framework to analyze from a systemic point of view the security guarantees provided by both, underlying infrastructure technologies and storage technologies commonly used in Data Grid sites, focusing in typical attacks that can be mounted on the data and meta-data. Our goal is to identify the security gaps and even redundant security features that may affect the proposed Data Grid scenario, so that ongoing research may be focused in fulfilling these needs while keeping a fair balance between security and performance.

The rest of this paper is organized as follows: section 4 introduces the Data Grid use-case to be analyzed along this paper, which is based on the OGSA Data Working Group's Data Architecture Scenarios [5]. Section 3 presents and extends the security evaluation framework from [6] so it can be applied for analyzing Data Grid's storage systems involved with the proposed use-case. In section 4 we summarize the security guarantees provided by state-of-the-art underlying Grid infrastructures. In section 5 is applied the extended framework to analyze the potential risks and security gaps of commonly deployed Grid storage technologies. Section 6 concludes about the results obtained by the analysis and outlines our future work based on these.

2 A typical use case for Data Grid's storage services

In this section we introduce the Data Grid scenario that will be used throughout this paper to demonstrate the usability of the proposed framework, by analyzing its security issues from the point of view of the storage services. The scenario considers typical operations over data and metadata (i.e. creation, storage, replication, discovery and retrieval). It is mostly based on an OGSA Data Working Group's [3] draft recommendation [5], which describes a set of scenarios of generic nature that provide illustrations of how the components and interfaces described there can be put together in a selection of typical data scenarios. That document provides the building blocks to deploy complex Grid applications, therefore allowing to us extrapolate quite easily the results obtained in the present paper to real-world Data Grids.

Using the architectural patterns from [5], figure 1 presents a typical use case for the Data Grid, where a user (Client 1) accesses a Rendering Grid Service to generate some data and then proceeds to replicate it to three different sites (including a long-term storage). Later, a second user (Client 2) processes the generated data through a Visualization Service, which is in charge of locating an appropriate replica and copying the requested data to a local storage service for better performance. A simplified view of this scenario scenario leads to the steps shown in tables 1 and 2. Notice that it is possible for all the elements and resources to be located in different institutions, as this is the Grid philosophy.

Table 1: Steps taking place in the Extended Use Case for Client 1

Step	Explanation
1	Client 1 submits a job that invokes an operation from the Rendering Service.
2	The Rendering Service generates data from the job's execution.
3	The Rendering Service needs to write the generated data to a replicated Grid storage system, this is requested to the Replication Service.
3b	The Replication Service interacts with the Registry Service to enter the data resource (containing details such as creation time and so forth) into the replica catalog.
4	The Replication Service contacts its corresponding Data Transfer Service to request the necessary data storages.
5	Data Transfer Service 1 copies the generated data between the Data Services (in this case the three of them). This process can be conveyed through any protocol supported by the Data Services, i.e. GridFTP or RFT.
6	Finally, each Data Service stores the requested data into its respective Storage Service, including long-term storage for certain result sets.

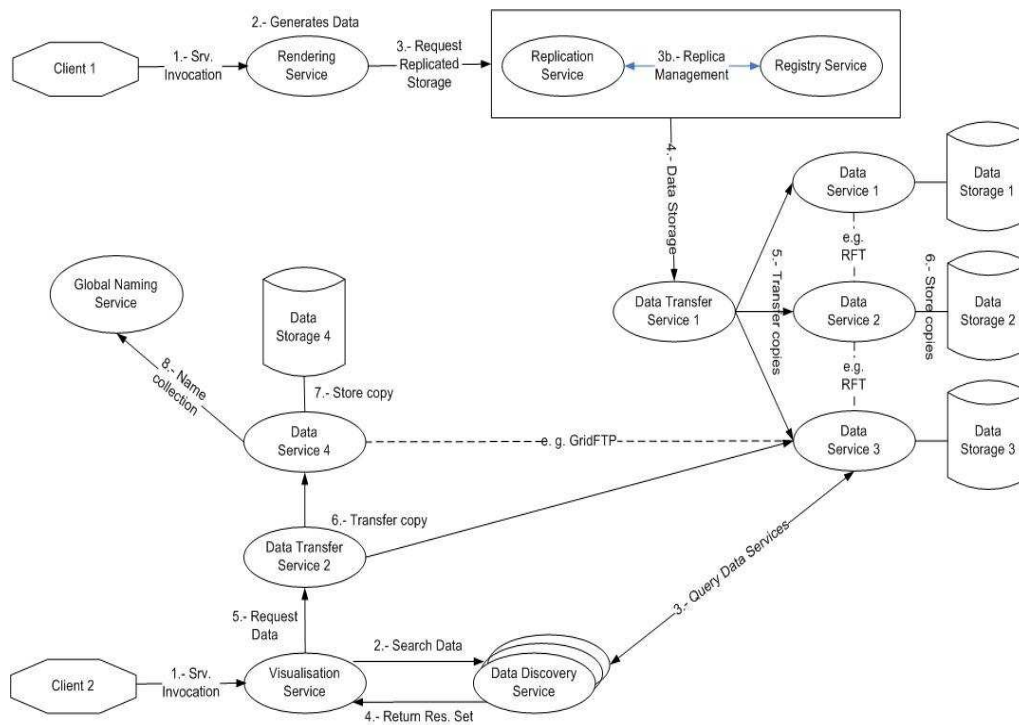


Figure 1: A typical Use Case for Grid Storage Services.

This use case comprises several security-related concerns; to analyze them we will extend in the next section a framework originally proposed for the security evaluation of storage systems in general.

3 An extended framework for evaluating Grid Storage Services security

The commonly accepted security services related with the use case presented in the previous section include Authentication, Confidentiality, Integrity, Authorization, Auditing, Privacy, Availability and Delegation. According to [28], this set of security services is defined in the following way:

- **Authentication (AuthN):** Defined as the process of verifying an identity claimed by or for a system entity. An authentication process consists of two steps: Identification and Verification.
- **Confidentiality:** Is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes. In most of the cases, encryption is the favorite mechanism to enforce confidentiality.
- **Integrity:** Known as the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner.
- **Authorization:** Refers to the process of granting privileges to processes and, ultimately, users. This differs from authentication in that authentication is the process used to identify a user. Once identified (reliably), the privileges, rights, property, and permissible actions of the user are determined by authorization.
- **Auditing:** Process for collecting data generated by services' activity, which may be useful in analyzing the security of a network and responding to security incidents.
- **Privacy:** In most of the cases, privacy is a reason for security rather than a kind of security just like happens with the Grids used in the biomedical arena [7].
- **Availability:** A security service that protects a system to ensure that it is accessible and usable upon demand by an authorized entity.

Table 2: Steps taking place in the Extended Use Case for Client 2

Step	Explanation
1	Client 2 submits a job that invokes an operation from the Visualization Service.
2	The Visualization Service needs the data generated from Client 1 (table 1), so requests its location from the Data Discovery Service.
3	Each one of the Data Discovery Service's nodes contacts its respective Data Service until the requested data is found. Notice that it is even possible to retrieve historical data from the long-term storage.
4	Once that data's location has been found, a result set with this information is returned to the Visualization Service.
5	The Visualization Service extracts the data's location from the returned result set, and instructs Data Transfer Service 2 to retrieve it.
6	Data Transfer Service 2 uses a supported protocol (i.e. GridFTP) to move the requested data from foreign Data Service 3 to local Data Service 4.
7	Data Service 4 uses Data Storage 4 to store the data.
8	The Data Transfer Service 2 contacts the Global Naming Service in order to name the Client 2's collection of data.

- **Delegation:** Is the process that allows a service to legitimately act on behalf of a user, for example to fetch data from a Grid resource.

From the point of view of the Data Grid scenario being studied, its subsystems may become attacked in several ways, however for the purposes of our research the framework proposed by [6] will be used and extended with specific Grid-related features to reflect the main concerns linked with data and meta-data security. To apply this framework we must determine five components: players, attacks, security primitives, granularity of protection, and user inconvenience. Next is provided a brief description of these component with a special focus on Grid storage:

- **Players.** All of the possible players that one has to consider for protecting stored data. Each player is listed with a set of legitimate actions it can perform. Any other action by that player is treated as an attack, and thus the player becomes an *adversary*. A list of players applicable to Grid storage systems is shown below:
 - a. Owners – create and destroy data, delegate read and write permission to other players, and revoke another user's read or write privileges on owned data.
 - b. Readers – read data once permission to read is delegated by owners.
 - c. Writers – modify data once permission to write is delegated by owners.
 - d. Wire – transfers data between other players. This refers to the WAN transfer of data in the Grid and not within each site.
 - e. Group servers – provides Grid-wide Authentication and Authorization (AA) services for other players.
 - f. Namespace servers – allow traversal of namespaces, such as provide support for lookup of stored files. An example is a file catalog providing Logical File Names (LFNs) and Storage URL (SURL) information.
 - g. Site services – All players inside the domain of a single site: storage servers, namespace servers, and the wire within a site. Site storage services include disk and/or tertiary storage (tape libraries) of data. We aggregate these into a single category, because they are controlled by a single administrative authority.

Using the previous concepts, the players involved with the use case presented in section 4 are shown in table 3.

- **Attacks.** The set of possible attacks cited in [6] depends on the architecture of each particular system. A fairly general set of possible attacks that can be mounted on a Grid storage system (data or metadata) follows:
 - a. By the adversary on the wire – for instance, an attack mounted on the protocol used to communicate files to the clients.

Table 3: Players participating in the Data Grid use case

Player	Entity from Use Case
Owners	Client 1 and Rendering Service
Reader	Client 2 and Visualization Service (reads data generated from Client 1)
Wire	WAN links implicit in figure 1.
Group Servers	Authentication (i.e. Certification Authorities) and Authorization (i.e. VOMS) servers -not shown in figure-.
Namespace Servers	Global Naming Service and Registry Service.
Site Services	Replication Service, Data Transfer Services, Data Services, Data Storage Services (including Long Term) and Data Discovery Service.

- b. By the adversary on the namespace servers – for instance, an adversary gaining SURL and/or LFN information.
- c. By a revoked user on the group servers – for instance where a revoked reader can continue to read files from tertiary storage.
- d. By the adversary colluding with the group server – for instance an adversary gaining access to AA services.
- e. By the adversary colluding with the site services – for instance, one where a filesystem directory is deleted by a user gaining access to Grid files by assuming a valid local user ID, i.e. a localized attack within a site that affects Grid data.
- f. By the adversary colluding with readers or writers – for instance, a reader passing a copy of a file to an adversary.

Each of the above attacks can be further classified into three classes based on their effects on the data:

1. Leak attacks: an adversary gains access to some data. Mostly related with the data’s confidentiality and the privacy.
 2. Change attacks: an adversary makes *valid* modifications to data. Data integrity is not tampered, even though its privacy is. The auditing service could be able to detect this kind of attacks.
 3. Destroy attacks: an adversary makes *invalid* modifications to stored data. Mainly linked with data integrity and availability, even though this attack may have been triggered via the site authentication and authorization mechanisms.
- *Core security primitives*. Secure storage systems implement several features to enable players to securely perform their functions. These primitives can be categorized into six types: authentication, authorization, securing data over the wire and on the disk, key management and revocation.
 - *Granularity of protection*. A system providing secure storage bears the additional overhead of cryptographic operations to support authentication, authorization, encryption of data over the wire and on the disk, key management and revocation. To limit the overhead of cryptographic operations, most systems implement different optimizations including aggregation of players into groups to simplify authorization, and trading off the security of short-lived keys against the ease of management of long-term keys.
 - *User inconvenience*. The level of inconvenience users are willing to tolerate before they become careless and prone to seek ways to circumvent security measures.

Applying the framework presented above, it is possible to identify potential damages caused by attacks that can be mounted on data or metadata from the Grid scenario introduced in section 4. A few examples of potential attacks are the following:

- An adversary in control of the replication service leaking the generated data from the Rendering Service (i.e. by moving it to the attacker’s site). This is even cheaper than trying to modify the underlying software to make an undetectable change attack.

- An adversary on the wire can perform a leak attack on the transmitted data (i.e. using a network packet sniffer); a more sophisticated change attack can through the WAN be mounted against other subsystems (i.e. man-in-the middle and session replay attacks to the GridFTP service).
- A revoked user colluded with a group server may be unnoticeable to the Visualization Service.
- An adversary with full access to site services (e.g. the Data Storage) may steal data from disks (leak attack, destroy attack) or re-write the stored data with a previous -and valid- version (change attack).

Damages caused by such attacks can be "combined", thus spreading the negative effects to every Grid subsystem. Our analysis views Grid Data's Security as a whole system, whereas if a subsystem is compromised then the negative effects are propagated to others. A property of this "systemic" view of the Data Grid's Security is that countermeasures can be applied to the *optimal* subsystem (i.e. where overall damage is reduced while performance is maximized), while the final effect is conveyed also to all of them. In the rest of this paper we will use the extended framework to analyze the overall security capabilities and gaps provided by both: the underlying state of the art Grid infrastructure, and a couple of widely used storage technologies.

4 Capabilities of the underlying Grid security infrastructure

Security in a Grid environment is all about layering. We can roughly divide the process to reach access to a Grid resource in 3 layers: first of all there is a Grid authentication process, then authorization on Grid-bases, and finally local enforcement using the resource-specific security capabilities. In figure 2 you can see the all-round security process. In the storage use case of section , we suppose that each client should first of all pass a Grid authentication

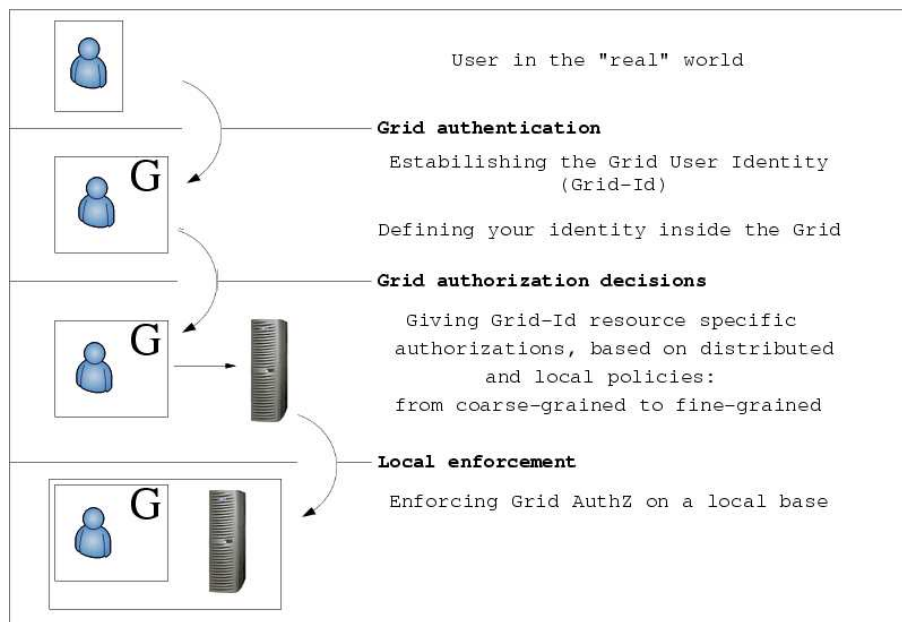


Figure 2: The security process

phase, that defines distinctly the Grid identity (Grid-ID) of any user: this means that every user inside a Grid is given a background, a description. With description we mean not only user's VOs, but his/her role inside every VO he is member of. Each Grid service takes part on the Grid authorization phase: for example, the Replication service, as well as each Data service, is a middleware element whose usage must be allowed by a Grid authorization service, normally identified with a policy framework. In figure 1, there are 3 Data Storage elements and a Long-term Data Storage: since each one of these storage elements may have a different local security implementations, being for example Kerberos or POSIX ACLs, there is the need for a local mapping, where Grid-authorizations are enforced using the local security implementation. Such a layering is used in section 4.3, where we analyze the security capabilities

of some implementations of the Grid Authentication and the Grid Authorization layers, and in section 5, where we consider the security infrastructures of some well-known Grid storage elements.

Next, we summarize the security capabilities offered by state of the art Grid-related infrastructure (technologies not collocated with the Storage service itself). In particular we will focus on the underlying Grid Security Infrastructure (GSI) and an important set of the so-called Authentication and Authorization Infrastructures, able to provide an overall security solution to current Data Grid installations. The reviewed infrastructures are summarized at the end of this section to show the protection they give against most of the attacks mentioned in section 3, and also in some cases we will show that the same security guarantees are offered by more than one technology. For our research it is important to find which of these security features are also implemented by the Grid Data Storage subsystem, i.e. to clearly identify not only gaps that may result in a potential attack, but also the redundant mechanisms that should be optimized to keep a balance between security and performance.

4.1 Grid Security Infrastructure

The Grid Security Infrastructure (GSI) [1] is comprised of a set of protocols, libraries, and tools that allow users and applications to securely access Grid resources. The Globus Toolkit [9] is a well-known and widely deployed implementation of GSI. Two mechanisms are defined by GSI: *Secure Conversation -Connectivity layer-* and *Secure Message security -Resource layer-*, for *authentication* and *secure communication (integrity and confidentiality)*. In the GSI Secure Conversation approach the client establishes a context with the server before sending any data. This context serves to authenticate the client identity to the server and to establish a shared secret using a collocated GSI Secure Conversation Service. Once the context establishment is complete the client can securely invoke an operation on the service by signing or encrypting outgoing messages using the shared secret captured in the context (through a SSL/TLS channel). The GSI Secure Message approach differs in that no context is established before invoking an operation. The client simply uses existing keying material, such as an X.509 end entity certificate, to secure messages and authenticate itself to the service. Securing of messages in the GSI Secure Conversation approach, i.e. using a shared secret, requires less computational effort than using existing keying material in the GSI Secure Message approach. This allows the client to trade off the extra step of establishing a context to enable more computationally efficient messages protection once that context has been established. For *authorization* purposes GSI uses the SAML standard [10] from OASIS. SAML defines formats for a number of types of security assertions and a protocol for retrieving those assertions even from third-party services, therefore enabling new features like role-based authorization. Finally, GSI provides also a *delegation* capability to reduce the number of times the user must enter his pass-phrase (single sign-on), this is performed through a *proxy certificate* [11].

4.2 Authentication and Authorization Infrastructures for Grids

As Virtual Organizations (VOs) increasingly turn into distributed multi-institutional collaborations, secure authentication and authorization become a growing challenge. VO membership may change dynamically, rights may be granted to entities on a periodic basis, or a users' role in an organization might dynamically evolve. Such factors make it more practical to express users' rights based on their other attributes, such as institutional affiliation or role in a collaboration, rather than identity alone. In Grid there is often a clear separation between the Certificate Authorities (CAs), which are the authorities of identity, and the authorities of attributes.

In the following we briefly review some of the authentication and authorization tools starting with authentication specific tools (VOMS and Shibboleth), then considering authorization specific tools (G-PBox, CAS, PRIMA and gPLAZMA) and concluding with the project PERMIS whose objective is to solve both the authentication and authorization issues.

The **Virtual Organization Membership Service (VOMS)** [2] is an *Attribute Authority* that exposes attributes and encodes the position of the holder inside the Virtual Organization (VO). A Holder may be a member of several groups, and may hold a special role inside some of his groups. Groups are organized in a tree structure that comprises groups and subgroups, while roles are not hierarchical and are associated to the membership in a group. The internal structure of a VO as defined in VOMS is not a full hierarchical Role Based Access Control (RBAC).

VOMS uses X.509 identity and proxy certificates and it is being extended to support SAML protocol and assertions. It works in *push* mode: the user, before contacting the Grid service, will contact a VOMS server to obtain user's Attribute Certificates (ACs) and then convey those to the target Grid service using X.509 proxy certificates.

Currently it is the most widely-used attribute authority service, and is a de-facto standard for AA in production Grids.

Shibboleth [35] is a system that asserts attributes about a user between organizations, i.e. between the user's home organization and organizations hosting resources that may be accessible to the user. It is both an identity and attribute authority, but it does not define any structure in the attributes it manages.

Shibboleth uses SAML assertions and it works in *pull* mode: it is the target Grid service that, after authenticating the user, will contact the appropriate Shibboleth service to obtain attributes information. It is primarily designed to work with Web applications and there is plans to support non-Web applications. There is the GridShib project [36] whose goal is to allow interoperability between the Globus Toolkit and Shibboleth.

The following tools are specific to authorization issues:

The **Grid Policy Box (G-PBox)** [38] framework is an approach for the management of policy repositories hierarchically distributed to independent, administrative-based layers, where each layer contains only policies regarding itself. It tackles the authorization problem using access control policies defined with the XACML [37] language. It is composed by a server, which is the Policy Decision Point (PDP) [15], and an administration Graphical User Interface (GUI). The PDP is a XACML (specification version 2.0) engine and Java, C++ and C libraries for Policy Enforcement Point (PEP) [15] communication are supplied. The GUI offers facilities for policy and distribution management, which means that it allows to create/remove/modify/move arbitrary XACML policies and policy sets and to send/receive policies to/from other PBoxes. G-PBox is a tool intended for authorization purposes and it relies on an external AA. In particular, it is already configured with a VOMS plug-in to retrieve user's Attribute Certificates from VOMS servers.

Community Authorization Service (CAS)[13] is an authorization service built on top of GSI. A user requesting access to a resource contacts the CAS server which, after a GSI-based authentication, issues a restricted proxy credential with an embedded access-control policy. The user utilizes this credential to connect to the requested resource, and then the resource itself applies its local policy to further restrict this access. With CAS, the ultimate decision about what happens at a Grid resource is removed from the resource provider and put in the hands of the CAS administrator. CAS does not record groups or roles, but only permissions.

Privilege Management and Authorization in Grids (PRIMA) system [14] is an authorization system that makes use of PDPs and PEPs interactions to focus on access control policies, while user attributes come from external Attribute Authorities like VOMS. This approach uses a VO-global specification of privilege attributes per Role, with local enforcement of privileges using the Grid User Management System (GUMS) [8] identity mapping service.

Grid-aware pluggable authorization management (gPLAZMA) [16] is a storage-specific security technology, developed for authorization within dCache [17]. After authentication, gPLAZMA uses PRIMA to query a storage authorization service that calls GUMS for local user mapping. It can operate in a role-based authorization mode.

Finally there is the **Privilege and Role Management Infrastructure and Standards Validations (PERMIS)** [12] project that focuses on both authentication and authorization issues. It is an implementation of a hierarchical RBAC mechanism that makes use of a powerful policy engine to take a policy file and then make authorization decisions based on this policy and the Attribute Certificates it receives. It relies on a non-standardized XML policy language and, since the roles are internally defined to every policy engine, its use is more appropriate for local sites rather than for VO-oriented environments.

4.3 Security capabilities as countermeasures

Table 4 summarizes the security capabilities provided by the infrastructure reviewed in this section. The layout of the table is based on the framework proposed by [6] and extended in section 3 to accomplish Data Grid-related needs. Results are categorized by possible attacks (main columns) and types of damage – the Leak, Change, Destroy sub-columns –. Cells marked with an “Y” mean that the system (row) prevents that type of damage caused by this particular attack. Cells marked with an “N” mean that the system does not prevent the damage. If a particular attack does not apply to a system, then the cell is marked with a “-”.

Our goal here is to highlight how this underlying infrastructure's security services can act as countermeasures for the typical attacks that can be mounted on the data and meta-data.

Table 4 shows that GSI mechanisms are common to all other infrastructure technologies, and therefore the attacks related specifically with the wire are prevented.

¹ Adversary with full control of the site services.

² See discussion about conditions required for this countermeasure.

Table 4: Security capabilities as countermeasures.

<i>Attack</i>	Adversary on the wire (WAN)			Adversary on the namespace servers			Revoked user on the group servers			Adversary with group server			Adversary with site services ¹		
	L	C	D	L	C	D	L	C	D	L	C	D	L	C	D
GSI + { VOMS, CAS, PERMIS, GPBox }	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
GSI+ PRIMA+ gPlazma	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y ²	Y ²	Y ²

Example 1. Leak attack This attack is prevented if GSI’s Secure Conversation is enabled (using encrypted channels), because the attacker will be unable to decode the plaintext data being moved between the players. On the other hand, there may be a significant *performance overhead* to consider if this mechanism is used to move massive amounts of data within the Grid.

Some other attacks can be also stopped with GSI+AAI technologies; consider for instance the following example.

Example 2. Attacker colluded with namespace server The attacker will be unable to perform any damage (change or destruction) to the data/meta-data, because GSI’s mutual authentication and authorization mechanisms based on cryptographic protocols prevent *untrusted* servers from interacting with Grid clients or other services. Note that only players coming from valid Certification Authorities are allowed to be members of a Virtual Organization. For an attacker to fully impersonate a valid namespace service, the service’s private key should be compromised first.

Regarding the attacks coming from compromised Group servers, even though they cannot be prevented by GSI or related technologies, in practice they are quite costly and difficult to execute. Consider the following example.

Example 3. Attacker colluded with group server Firstly, the potential attacker must compromise or collude with one or more Certification Authorities or Authorization Servers to impersonate a player and therefore perform some damage to the data/metadata. If we consider that Grid Certification Authorities fulfill a minimum set of security requirements (called *Authentication Profile* in the case of Grid Federations [34]), this attack is practically unfeasible, especially compared with the others explained in this section.

In the case of the site services and in particular with the storage service subsystem, if it is *GSI+AAI aware* then it will be protected from a wide variety of attacks due to the *inheritance* of their security features. If the site services implement additional security mechanisms then these redundant features may result in negative effects, just like mentioned in the following example:

Example 4. Redundant security features - gPlazma Let us consider the case of gPlazma, where the use of an additional authorization mechanism at the site service might result in performance degradation when re-checking the GridID’s permissions also at this level (additional to the VO-level authorization mechanism).

Under our assumption of an attacker with *full* control of the site services, the surveyed underlying technologies are not enough to protect against the attacks mentioned in table 4. The following example may help to clarify this point.

Example 5. Attacker colluded with site services - full control In the case of an attacker with *full* control of the site services³ then even gPlazma and their underlying technologies may not be enough to protect against data leaks (the data into the storage device is not encrypted, therefore the attacker may access it), data changes (re-writing stored data with older, but still valid information that will pass all the integrity checks) and destruction (low-level formatting the device or physically damaging it).

Even though there may never be a *total-security solution* for untrusted storage sites, the security concerns related with these are the entry point for the next section, where state of the art storage technologies are analyzed within the extended framework presented in section 3 as a way to provide overall conclusions and research ideas about its security guarantees and gaps.

5 Security capabilities of Grid Storage Systems

Before investigating security issues in Grid storage systems, it is useful to introduce the existing Grid storage model and in which way the storage systems are accessible and manageable from the Grid.

In Grids for distributed computing and data access, the resources are not only geographically dispersed and spanned over multiple trust domains, but they are also composed of a variety of heterogeneous hardware and file-systems. Moreover they might differ by the quality of service offered, by the semantics for data access - both for reading and writing - and by interfaces or security mechanisms implemented. Therefore, one of the main goals of the Grid consists in providing a way to share all these resources by defining a description of the storage resource at the conceptual level, that is, an information model representing an abstraction of the real storage resources. Also quite important is the implementation of a common interface to users, regardless of the type of the backend system being used.

The Storage Element (SE) is the core concept of the model mentioned above, named GLUE schema [40], and identifies the group of services responsible for the storage resource. At the virtual level, the storage resource is abstracted using the concept of storage area that can be made accessible to groups of users or VOs by using the *AccessControlBaseRule* attributes whereas the entities *AccessProtocol* and *ControlProtocol* are used to publish endpoints of protocols related to the storage resource.

The *AccessProtocol* describes allowed ways to transfer files to and from an SE, like GridFTP and RFIO (an HEP-specific protocol) and access files which implies direct Posix access (e.g. NFS, AFS, GPFS, LUSTRE). Current *ControlProtocols* include the SRM (Storage Resource Manager) [18] that provides a fully transparent and dynamic management to underlying storage resources. The SRM paradigm is nowadays a building-block of the storage requirements in the computing Models of the HEP experiments, aiming to provide the dynamic management of a storage resource engaged in a distributed computing system. SRM services agree on a standard interface to hide storage dependent characteristics and to allow interoperability between different storage systems. The implementation of storage management services, compliant to the same version of the SRM protocol, is the basic step to provide an infrastructure able to inter-operate in a transparent way.

Current state-of-the-art Grid storage systems are hierarchical storage management systems (HSM), managing storage distributed on disk and tape libraries, and disks only storage system, based on cluster and parallel file system to manage and aggregate the storage on many distributed nodes.

In the Data Grids the most widely used grid storage systems are: *dCache* with *PNFS* [17], *DPM* [21], *CASTOR* [24], *STORM* [22] and *SDSC SRB* [29]. These can be categorized as systems offering storage distribution across more than one grid sites (*dCache*, *DPM*, *SDSC SRB* and *StoRM*) and systems operating within a single site (*CASTOR*). Since our analysis focuses on grid distributed storage, we will present in more detail the capabilities of the former class of systems, however a brief analysis for *CASTOR* will be also introduced for completeness. We must make it clear that these are not directly comparable to *dCache*, *DPM* and *StoRM* and thus will not be included in the capability table of section 5.6. This report complements previous works that provided an overview of functional capabilities [30] and a performance evaluation [31]

Our goal in this section is to apply the extended security framework defined in section 3 as a tool to determine the potential set of attacks feasible to be performed against the Data Grid storage systems cited here. It is important to note that the applied framework is not intended to allow evaluation of the end-to-end security of a particular system. This requires careful analysis of each component and the particular way of combining them, since any secure system is only as strong as its weakest link. The framework is neither intended as a replacement for such analysis, but simply seeks to allow a high-level comparison among different systems, purposely leaving some secondary details unexamined.

³This is the case of untrusted sites, which become a serious security problem for the Data Grid.

Next we present a brief outline of each system including its security capabilities, followed by a table summarizing the potential attacks and damages over each one of these.

5.1 dCache

dCache/SRM [17] is a grid storage middleware system that combines distributed heterogeneous disk storage systems under a single filesystem tree. It also handles data hot spots, hardware faults and replication for availability purposes.

To provide Grid functionalities (compliant with definitions of the LHC computing Grid (LCG) Storage Element storage fabric) dCache supports:

- A protocol for local access of data. For performance and decentralization reasons dCache uses PNFS [19], an NFS-like service which allows namespace operations to be performed through a standard NFS2 interface, while data transfer is performed through faster channels via a number of protocols, including a native dCap protocol (DCCP), GridFTP, and HTTP. PNFS is not a filesystem designed for storage of actual files but instead it manages the filesystem hierarchy and standard meta-data of a UNIX filesystem. dCache uses PNFS to store its meta-data. PNFS implements an NFS server, however it is different in many ways and cannot coexist with other NFS servers. All the meta-data can be accessed with a standard NFS client, like the one in the Linux kernel. After mounting, normal filesystem operations work fine. However, I/O on the actual files in the PNFS will normally result in an error. Data transfer operations with PNFS are fast, however, namespace operations are still handled by a central component (database) and are much slower. Performance is expected to improve when the name space module is replaced by a new database system, Chimera.
- A secure wide-area transfer protocol, which currently is GsiFTP (GridFTP). dCache allows opening files using a URL-like syntax without the PNFS filesystem being mounted. This is done through one of the supported protocols (dCap, GssFTP, GsiFTP and HTTPS).
- Each storage element needs to provide status information (availability, load, free space). This is currently done using LDAP.
- A protocol to make storage area manageable (SRM interface). Besides namespace operations, it allows to prepare data sets for transfers directly to the client, as well as to initiate 3rd-party transfers between SEs. SRM retries failed transfers and also handles space reservation and management. Moreover, it protects storage systems and data channels from overload by scheduling transfers appropriately. SRM does not execute transfers directly, but allows negotiation of the transfer protocols by the data exchanging parties.

Applications use dCache by linking a user-level library providing POSIX-like file I/O calls. This library supports pluggable security mechanisms, where GssApi (Kerberos) and SSL modules have already been implemented.

dCache can also be connected to tertiary storage systems, using custom protocols dependent on the tape system used on every site. Furthermore, it supports autonomous data distribution to various pools using rules/preferences. (e.g. all incoming data are first stored on high-perf disks and later flushed to tape storage). More complex setups consider load-balancing and other factors. There is a file replica manager, that keeps several replicas of each file on different pools according to site or grid-level policies.

On any dCache implementation, a node acts as the "admin" node, also known as the dCache server. This node runs PNFS and a number of other dCache services. In addition to the admin server, many file pools may be added. The pool nodes are where files are stored. There is no problem having NFSd or GridFTP running on a pool node.

5.1.1 Security Framework Characteristics

Players The following four are involved:

- The PNFS daemon on the admin node is the namespace server.
- Pool nodes are the storage servers.
- Door nodes are the group servers providing authentication and authorization based on GSI and VOMS services.
- Replica manager daemons handle file replication to various storage pools.

Trust Assumptions Seven assumptions were considered:

- Storage servers are authenticated to admin node running PNFS and to clients using dCap or GssApi, but they are trusted with the data (same for disk or tertiary storage). Also dCache is not protected from attacks in collusion with the site services.
- Data are protected on the wire to the clients and through the WAN using GssApi or SSL. However, namespace protocol (NFS2) data are unprotected on the wire. Upcoming versions will use NFSv4, which can be built on top of a secure RPC framework (using a plugin architecture). Furthermore, file data are not protected on the wire within a single site network, where the NFS protocol is used for namespace operations and local file access.
- Data are not protected on disk or tapes.
- Since all authentication and authorization data are present in the group servers, dCache is vulnerable to attacks in collusion with the group server on the door nodes.
- Certificates for access to files are currently being issued for a large time window (e.g. 12 hours) in most implementations. This may allow a revoked user to retain access rights to files within this window if real-time validation is not performed, just as recommended in [20].
- The namespace servers map file names and attributes to data on storage pools only available to clients with valid certificates.
- Since the replica manager has copy, move and delete access to files on storage nodes, dCache is vulnerable to attacks in collusion with the replica manager.

Security Primitives We have taken the following:

- dCache provides create/read and delete access to a file, it does not allow the modification of a file. An existing file would have to be deleted and written back. However the storage server or the metadata server (admin node running PNFS) may read, modify or delete files on storage nodes.
- The group server authenticates and authorizes clients using the GSI and VOMS infrastructure.

5.2 DPM: Disk Pool Manager

The Disk Pool Manager (DPM) [21] has been developed at CERN as a disk-only Storage Element, supporting the SRM-compliant Storage Element (SE) interfaces, without being complicated by other modes of access or complications such as tape storage systems. DPM as an alternative to the classical SE offers the following advantages: (i) SRM Interface, (ii) Better scalability (allows the management of 10+TB distributing the load over several servers), (iii) High Performance and (iv) Light-weight management. DPM relies on a MySQL database to store its metadata.

5.2.1 Security Framework Characteristics

Players Three players were considered in the analysis

- A DPM daemon offers access to a set of filesystems, located on one or more file-server nodes.
- A SRM service supports the protocol for Grid access to the storage resources, by translating requests to the native DPM protocol.
- A name service, built over a MySQL database, supports namespace lookup operations.

Trust Assumptions Our study considered the following:

- Site services are trusted if clients pass through the authentication and authorization services (based on GSI and VOMS) and hold valid certificates.
- Data are protected on the wire to the clients and through the WAN using GssApi or SSL. Within a single site network data are not protected on the wire.
- Data are not protected on disk or tapes.
- Certificates for access to files are currently being issued for a large time window (e.g. 12 hours) in most implementations.

Security Primitives Two main primitives exist:

- DPM allows create/read and delete access to files, but does not allow direct file modification. However the storage and namespace servers within a site may read, modify or delete files local files.
- Authentication and authorization in DPM are based on the GSI and VOMS services, described in section 4. Grid IDs are then mapped to local user IDs which are in turn used to access files as local users.

5.3 SDSC SRB

San Diego Supercomputer Center's Storage Resource Broker (SDSC SRB) [29] is a client-server architecture, that exports a unified view and authorization mechanism of the available datasets while hiding their actual location. These data resources may be heterogeneous in nature, distributed along multiple hosts and platforms. Users can store or replicate their data collections across several servers, while maintaining total access control locally. A Metadata Catalog (MCAT) service supports user queries related to datasets and their properties, including a mapping from logical handles to physical file locations. Long-term preservation and data provenance are important design goals for the SDSC SRB. The SRB system is middleware in the sense that it is built on top of other major software packages (file systems, archives, real-time data sources, relational database management systems, etc).

5.3.1 Security Framework Characteristics

SRB's security framework provides a mechanism by which data files can be encrypted for both network transmission and storage, contrary to GSI which only provides a secure communication channel. A key idea for this system is to encrypt (and/or compress) on the client side and store the files in that form; this is performed thanks to SRB's use of OpenSSL [33] libraries. With this mechanism the client performs a similar amount of work as it would if GSI had been used, but on the server side it requires very little additional work.

For authentication purposes, SRB may use GSI mechanisms besides its proprietary Encrypt1 system [32]. On the other hand, a proprietary mechanism (MCAT based) is used for authorization.

Players Three players are involved in a typical SRB session:

- The SRB Client that reads and writes the data.
- The SRB Server that process requests by *(i)* accessing its storage elements, or *(ii)* communicating to remote SRB Servers.
- MCAT (Metadata Catalog), stores the metadata and translates from logical to physical data representations.

Trust Assumptions The following six were used for our analysis:

- The User authenticates to the Data Grid (GSI or SRB Server act as Group Servers), which in turn performs authorization (MCAT based).
- For secure communication SRB relies on its own mechanism (SRB Secure Communication or SSC), instead of GSI's.
- SRB Servers authenticate among themselves if necessary.
- Each remote storage system must create an account ID under which the SRB Server stores files. Whatever storage resource is used, the SRB server process interacts with it as a client and authenticates itself as a particular user
- For the SRB system to be secure, the operating systems on which it runs must also be secured, in particular with the hosts.
- The whole SRB system is only as secure as the MCAT (i.e. DBMS system). The MCAT contains data about user names, group memberships, access rights, user and group passwords, resource descriptions, locations (network hostname mappings), the entire logical/physical namespace (mappings of each individual data file from its logical name to physical locations and names), etc. Moreover, for encrypted data if the MCAT is lost, so is the data.

Security Primitives Three security primitives are used:

- The group server authenticates (using GSI or SRB Servers) and authorizes (only through the MCAT) SRB Clients.
- All stored data is encrypted, so any operation over it (read/write/delete) requires decryption on the SRB Client. This means that data on the wire is secured.
- A random key is generated for encrypting each data file and this key is securely transferred to (and later from) the MCAT. It would be possible to retrieve encrypted files without an MCAT, if the file key is available. SRB admins can utilize DBMS features to safeguard the MCAT. Users can also record the key with supplied SRB client software.

5.4 StoRM

The Grid Storage Resource Manager, in short StoRM [22], is an implementation for disk-based storage system of the version 2.2 of the SRM interface, developed at the INFN - CNAF centre in collaboration with the EGRID [42] project founded by the ICTP centre. It is designed to satisfy the requirements coming from the HEP experiments, in particular those installed at the LHC, together with the requirements coming from the financial Grid use cases.

The main idea behind StoRM is to create a SRM implementation to leverage on the advantages of high performance parallel file-systems in a Grid environment. It provides users and applications with the standard SRM functionalities combined with the capability to perform secure and local accesses to the shared storage resources (e.g. through GPFS).

StoRM provides the advanced SRM functionalities to dynamically manage space and files according to the user requirements, to specify the desired lifetime, allow for advance space reservation and different qualities of service provided by the underlying storage system. All these functionalities are implemented in StoRM allowing a direct file-system access to the files, without interacting with any external data-access service. In this way local accesses and Grid accesses can coexist on the same storage resource, and non Grid-aware applications can be easily brought into the Grid environment without any modification in the data access pattern.

StoRM has a multi-layer architecture made by two main components: the *front-end*, that exposes the SRM web service interface and manages user authentication, and the *back-end*, that is the core of the StoRM service, with a database system used to store SRM requests and the StoRM metadata. Support for the different file-systems is provided in StoRM by a driver mechanism. The back-end logics is decoupled from this wrapper component and the specific driver can take advantages from proprietary functionalities provided by certain file-systems. The driver

defines a common internal interface for the underlying file-system in use, allowing a StoRM instance to work on different file-systems at the same time, such as GPFS [23], Lustre [39] or XFS [41].

For managing and aggregating the storage resources StoRM takes advantage from dedicated systems, like the General Parallel File System (GPFS). GPFS is a high-performance shared-disk cluster file-system developed by IBM. GPFS distinguishes itself from other cluster file-systems by providing concurrent high-speed file access to applications executing on multiple nodes. With GPFS, a single file-system containing several PB of data can be built on dedicated hardware in a Storage Area Network (SAN) configuration. In a large farm it is not feasible to connect each worker node in the GPFS cluster directly to the SAN. For this reason GPFS, making use of a capability called NSD (Network Shared Disk), provides a sort of software simulation of the SAN via Gigabit network. In such a configuration, data to the I/O servers flows over the SAN and both data and control information to the clients flow across the LAN. The capability to build cluster using a LAN connection, available in GPFS as in other cluster file systems, allows StoRM to manage storage resources distributed across more than just a single grid site.

5.4.1 Security Framework Characteristics

Since it generalizes the file-system access to the Grid, StoRM also take advantage from the file-system security mechanisms to ensure an authenticated and authorized access to the data. StoRM provides a layered and flexible security framework to satisfy the requirements coming from the different scenarios involved.

- User authorization is based on X.509 proxy certificates and it supports the groups and roles attributes as defined by the VOMS service
- To verify if a user is authorized to perform a certain operation on the data StoRM is able to interact with external authorization catalog and services, as the G-PBox authorization tool.
- To provide a local secure access to data, POSIX ACLs are enforced on the desired files and directories with the local identity corresponding to the (mapped) user credential.

Players Three players were considered in the analysis

- A cluster file system that builds a standard file system providing access to distributed data across the cluster's node.
- A SRM service that provides the SRM functionalities for the storage resource management.
- At site level, GPFS/POSIX FS nodes are the namespace servers and storage servers, storing files and enforcing authorization for physical files using ACLs with local IDs.

Trust Assumptions Our study considered the following:

- Clients are authenticated and authorized by the StoRM front-end using GSI over HTTP.
- Site services are trusted if clients pass through the authentication and authorization services (based on GSI and VOMS) and hold valid certificates.
- Data are protected on the wire to the clients and through the WAN using GssApi or SSL.
- Since grid identities in StoRM are mapped to local users at site level in order to access the local files, StoRM is vulnerable to attacks in collusion with a site's group server and/or a site's storage server running the filesystem.
- Data are not protected on disk or tapes, because GPFS or other cluster file system does not by default encrypt data.

Security Primitives Three main primitives exist:

- StoRM authentication is based on GSI and VOMS service, and authorization information can be retrieved by local information or querying external authenticated services or catalog.
- To authenticate and authorize users in the VO domain, StoRM uses the Grid Security Infrastructure (GSI) for authentication of users and an external authorization services to retrieve authorization information.
- If authentication and authorization are okay, StoRM retrieve the mapping on grid identities to local user IDs and enforce a physical ACL with these local `group_id` and `user_id` on the files and directory on the filesystem to allow physical file access.

5.5 Systems for a single site

In this section we present a brief analysis for CASTOR only for reasons of completeness, because this is more suitable for operating withing a single site.

5.5.1 CASTOR

The name stands for the CERN Advanced STORage manager [24] and is a hierarchical storage management (HSM) system developed at CERN and used to store their produced files and user files. These can be stored, listed, retrieved and accessed in CASTOR using command line tools or applications built on top of the different data transfer protocols like RFIO (Remote File IO), ROOT libraries, GridFTP and XROOTD. CASTOR manages disk cache(s) and the data on tertiary storage or tapes.

CASTOR provides a UNIX-like directory hierarchy of file names. The directories are always rooted `/castor/sitename` (e.g. `/castor/cern.ch` for the CERN site). The CASTOR namespace can be viewed and manipulated only through CASTOR client commands and library calls. The namespace holds permanent tape residence of files, while the more volatile disk residence is only known to the *stager*, which is the disk cache management component. Accessing or modifying files in CASTOR always requires the use of a stager.

The main functionality can be grouped in 5 modules, briefly introduced as follows:

- The Stager, a disk pool manager whose role is to allocate space on disk to store files, to maintain a catalog of all the files in its disk pools and to clear out old or least recently used files in these pools when more free space is required. A disk pool is simply a collection of file systems.
- The Name Server, whose role is to implement a hierarchical view of the CASTOR name space so that it appears that the files are in directories. Names are made up of components (between `//`) and for each component, file access permissions, file size and access times are stored. The name server also remembers the file location on tertiary storage if the file has been migrated from the disk pool in order to make space for more current files. Files may be segmented or be made up of more than one contiguous chunk of tape media. This allows the use of the full capacity of tape volumes and permits file sizes to be bigger than the physical limit of a single tape volume. Additionally, it provides the ability to create directories and files, change ownership etc as specified in the POSIX standard.
- Tertiary storage, high performance cartridge tapes are used for tertiary storage in Castor, housed in Libraries or Robots from various vendors. The Castor Volume Manager database contains information about each tape's characteristics, capacity and status. The Name Server database mentioned above contains information about the files (sometimes referred to as segments) on a tape, ownership and permission information and the file offset location on the tape. User commands are available to display information in both the Name Server and Volume Manager databases.

The mounting of cartridges to and from tape drives is managed by the Volume Drive Queue Manager (VDQM) in conjunction with library control software specific to each model of tape library. The cost of tape storage per Gigabyte is still a lot less than hard disk, and it does have the advantage of being considered "permanent" storage. However, access times for data seen by users are of the order of 1-10 minutes rather than 1-10 seconds.

- The database, which plays a central role in the CASTOR design. Its database schema is very complex and contains about sixty tables and several procedures and triggers. The functionality of the database is to store the actual status information of ongoing processes to have stateless components.
- The Client allows you to interact with the server in order to get the basic functionality. You can get a file which is stored in the disk server or tape server, using RFIO, ROOT, GRIDFTP or XROOTD. You also can check the status of a file or update it, as well as put a file. The client can be CASTOR client (command line mode or API) or SRM.

The Storage Resource Management SRM is a middleware component for managing shared storage resources on the GRID (enlace a grid). It provides dynamic space allocation on a file management and uniforms the access to heterogeneous storage elements, CASTOR is one of them.

The design of CASTOR is modular, with a central database for information handling. Regarding storage security, confidentiality of the data transfer was not considered a design requirement, as CASTOR was originally developed for the High Energy Physics community. Moreover, it was assessed that file encryption would significantly impact performance and raised the issue of key management. Another important design decision was to plan for integrity checks for the overall system at a later stage (based on checksums stored in the CASTOR Name Server) rather than on a per-connection basis [25, 26].

Finally, the CASTOR Security library (Csec) has also been developed to support plugin modules as a means to provide string authentication. This library has been integrated in many components of CASTOR [27].

5.6 Security Analysis Table

Using the same notation from table 4, we analyze in table 5 both DPM and dCache in terms of their resilience to potential attacks.

Table 5: Summary of security guarantees provided by Grid Storage Systems.

<i>Attack</i>	Adversary on the wire (WAN)			Adversary on the namespace servers			Revoked user on the group servers			Adversary with group server			Adversary with site services ⁴		
	L	C	D	L	C	D	L	C	D	L	C	D	L	C	D
dCache	Y	Y	Y	N	N	N	Y ⁵	Y ⁵	Y ⁵	N	N	N	N	N	N
DPM	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N	N	N	N
SDSC SRB	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N	Y	N	N
StoRM	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N	N	N	N

Thanks to the use of the underlying GSI+AAI mechanisms, it is possible to inherently offer security features to the Grid Storage systems surveyed in table 5 just as mentioned in the next couple of examples:

⁴Adversary with full control.

⁵Within period of issued file certificates.

Example 6. Adversary on the WAN In the case of attacks directed to the WAN (passive attacker intercepting network traffic) these can be alleviated thanks to the *inheritance* of GSI+AAI's security features, in particular the use of authenticated and encrypted communication channels.

Example 7. Revoked user on the group server Attacks performed by a revoked user on the group servers can be prevented if GSI's GridID validation features are used (i.e. to verify that the digital certificate is not revoked on every operation). Obviously, also GSI's security flaws are inherited: surveyed technologies cannot prevent the damages coming from attackers colluded with group servers.

As discussed earlier, despite dCache and DPM systems might rely on GSI+AAI technologies, it should be noted that using *non-GSI* aware subsystems opens the door for attacks directed to namespace servers. Let us take the following example:

Example 8. Attacking non-GSI aware subsystems Since the namespace services implemented by the surveyed technologies (PNFS in the case of dCache and MySQL for DPM) are not using GSI's security features, they do not offer protection against attackers. Something similar occurs with SDSC SRM, where the documentation does not specify the integrity mechanisms offered to the data on the WAN if only the SSC protocol is being used instead of GSI.

Finally, as highlighted in the previous section, untrusted storage sites result in a critical security flaw for the Data Grid:

Example 9. Adversary colluded with site services Contrary to dCache and DPM which assume that site services are *trustworthy* -and are therefore quite sensitive to untrusted sites-, SDSC SRM encrypt stored files so leak attacks coming from adversaries colluded with site services are more difficult to execute. There are however a couple of weak points in SDSC SRM: (i) the MCAT itself can be compromised and (ii) once the storage element has the encryption key from the MCAT, this may be obtained (i.e. through a memory dump) by an attacker in full control of the site.

Due the importance of studying the security issues related with Data Grid's untrusted storage sites, the next section will mention our future research directions towards preventive measures against attacks related with site services.

6 Conclusions and future work

This paper presents the first part of our research on Grid storage system's security, analyzing the security of state-of-the-art technologies using a framework originally proposed for generic storage systems which we have extended to Grid-specific configurations. Our goal is to describe the potential set of attacks on the data or the meta-data along with the guarantees currently provided by the surveyed technologies, as a way to find out not only the gaps that must be covered, but also the mechanisms that should be optimized due to the redundant security services they are providing at the different Grid layers (i.e. authorization decisions taken by both, local storage systems and Virtual Organizations).

Most of Data Grid's storage technologies rely on, or inherit their security features, from the underlying Grid Security Infrastructure (GSI) and generally only need to call-out a third-party service for enhanced authorization decisions (e.g. G-PBox with VOMS). GSI provides a wide set of base security capabilities, but still is susceptible to attacks where the adversary has taken control of the group servers; however this is an unlikely scenario because most Certification Authorities and Authorization servers implement strong security mechanisms for their own protection.

Before the analysis our belief was that encryption functionalities were redundant at least in two layers of the Data Grid (the Wire and the data at rest), however resulted that only in the case of gPlazma and G-PBox the authorization mechanism was also implemented at the storage element. This feature may provoke some performance issues, but it has showed also the importance of studying untrusted storage sites because in this case *none* of the surveyed technologies are capable of providing an adequate protection. As an example let us focus in a scenario where the attacker has taken control of the storage device itself: even though low-level encryption may be enough to protect the assets, what happens with the encryption capabilities provided by upper layers like GSI? Is it feasible to re-use them at the lower-layers to improve performance?

Having identified the security gaps related with untrusted site services, our future work will focus on the mechanisms able to manage them while keeping a fair balance between performance and security. In particular we have chosen two approaches for this research:

1. Encryption based: Secure data at rest, but managing the encryption keys in such a way that untrusted storage elements and/or compromised clients can not decrypt them (i.e. when colluded). Performance is also an issue in this approach.
2. Policy based: This is mostly a preventive approach, where each storage element is identified by an advertised security policy that can be quantitatively associated to a security level. Clients are provisioned of storage resources based on a security criteria that considers this level.

Acknowledgments

We would like to thank Christos Papachristos and Panos Chatziadam for providing us with detailed information about the configuration of storage systems in the context of the HellasGrid Task Force (infrastructure that is part of the EGEE-II: Enabling Grids for E-scienceE project).

References

- [1] V. Welch. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective. The Globus Security Team. 2005. <http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf>
- [2] EU DataGrid, VOMS Architecture v1.1. March, 2007. <http://grid-auth.infn.it/docs/VOMS-v1.1.pdf>
- [3] OGSA-Data Working Group (OGSA-D-WG). March, 2007. <https://forge.gridforum.org/sf/projects/ogsa-d-wg>
- [4] Trust and Security in CoreGRID. April, 2007. <http://www.coregrid.net/mambo/content/view/281/275/>
- [5] D. Berry, et. al. OGSA Data Architecture Scenarios - version 0.15. March, 2007. <https://forge.gridforum.org/sf/go/doc14073?nav=1>
- [6] E. Riedel, M. Kallahalla, R. Swaminathan. A framework for evaluating storage system security. In *Proceedings of the 1st Conference on File and Storage Technologies (FAST)*, Monterrey, CA, USA, January 2002.
- [7] BELIEF: Bringing Europe's eElectronic Infrastructures to Expanding Frontiers. March, 2007. <http://www.beliefproject.org/>
- [8] GUMS – The Grid User Management System. April, 2007. <http://grid.racf.bnl.gov/GUMS/index.html>
- [9] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In Springer-Verlag LNCS 3779, *IFIP International Conference on Network and Parallel Computing*, pages 2-13, 2005.
- [10] Security Association Markup Language (SAML) Specification v.1.0. April, 2007. <http://www.oasis-open.org/committees/security/>
- [11] S. Tuecke, et. al. Request For Comments 3820: Proxy Certificate Profile. Network Working Group, June 2004. <http://www.ietf.org/rfc/3820.txt>
- [12] D. Chadwick, O. Alexander. The PERMIS X.509 Role based privilege management infrastructure. In ACM, *SACMAT '02: Proceedings of the 7th ACM symposium on Access control models and technologies*, pages 135–140, Monterey, California, USA, June 2002. ACM Press
- [13] L. Pearlman, et al. A Community Authorization Service for Group Collaboration. In IEEE, *Proceedings of 3rd International Workshop on Policies for Distributed Systems and Networks*. 2002. IEEE Computer.

- [14] M. Lorch, et. al. The PRIMA system for privilege management, authorization and enforcement in grid environments. In *Proceedings of the 4th International Workshop on Grid Computing*, Nov. 2003.
- [15] J. Vollbrecht, et. al. Request For Comments 2904: AAA Authorization Framework. Network Working Group, August 2000. <http://www.ietf.org/rfc/rfc2904.txt>
- [16] A. Rana. gPLAZMA : Introducing RBAC Security in dCache. In *Computing in High Energy and Nuclear Physics 2006*.
- [17] P. Fuhrmann and V. Gu:izow. dCache, storage system for the future. In *Europar 2006*, Dresden.
- [18] A. Shoshani, A. Sim and J. Gu. Storage Resource Managers: Essential Components for the Grid. In *Grid Resource Management: State of the Art and Future Trends*, 2003. Kluwer Academic Publishers.
- [19] Perfectly Normal File System (PNFS). <http://www-pnfs.desy.de/>
- [20] J. Luna, O. Manso and M. Medina. Using OGRO and CertiVeR to improve OCSP validation for Grids. In Springer-Verlag, *Journal of Supercomputing: special issue Technology Deployments in Grid Computing*. Netherlands, March 2007.
- [21] Disk Pool Manager. May 2007. http://www.gridpp.ac.uk/wiki/Disk_Pool_Manager
- [22] E. Corso, et. al. Storm, an SRM Implementation For LHC Analysis Farms. In *Computing in High Energy and Nuclear Physics (CHEP 2006)*, Feb. 2006.
- [23] F. Schmuck and R. Haskin. GPFS: A Shared-disk File System for Large Computing Centers. In *USENIX Conference on File and Storage Technologies*, pages 231–244, Monterey, CA, Jan. 2002.
- [24] O. Ba:rring, et. al. Storage Resource Sharing with CASTOR. In IEEE, *Proceedings of NASA Goddard 21st IEEE Conference on Mass Storage Systems and Technologies (MSST2004)*, Apr. 2004.
- [25] B. Couturier. CASTOR Security. Nov. 2005. http://castor.web.cern.ch/castor/docs/guides/dev/security/CASTOR_Security_Plan_2006.pdf
- [26] B. Couturier and V. Motyakov. Implementation of Strong Authentication in CASTOR. May 2007. http://castor.web.cern.ch/castor/docs/guides/dev/security/CASTOR_Security_Implementation.pdf
- [27] CASTOR. May 2007. http://castor.web.cern.ch/castor/DOCUMENTATION/CODE/SECURITY/CASTOR_Security_Implementation.pdf
- [28] R. Shirey. Request For Comments 2828: Internet Security Glossary. Network Working Group, May 2000. <http://www.ietf.org/rfc/rfc2828.txt>
- [29] C. Baru, R. Moore, A. Rajasekar and M. Wan Michael. The SDSC Storage Resource Broker. In *Proceedings of the 1998 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, Toronto, Canada, pages 5–17, 1998.
- [30] G. Stewart, D. Cameron, G. Cowan and G. McCance. Storage and Data Management in EGEE. In *Proceedings of Conferences in Research and Practice in Information Technology*, Volume 68, pages 69–77, 2007.
- [31] G.A. Cowan, G. Stewart, and J. Ferguson. Optimization of Grid Enabled Storage at Small Sites. In *Proceedings of 6th UK eScience All Hands Meeting*, Paper Number 664, 2006.
- [32] Encrypt1. May 2003. http://www.sdsc.edu/srb/index.php/Secure_Compressed_Data
- [33] OpenSSL. Feb 2007. <http://www.openssl.org>
- [34] IGTF: the International Grid Trust Federation. June 2007. <http://www.gridpma.org/>

- [35] Von Welch, Tom Barton, Kate Keahey and Frank Siebenlist. Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration. In *Proceedings of the 4th Annual PKI R&D Workshop*, pages 19–21, April 2005.
- [36] GridShib Project Web site. <http://gridshib.globus.org>
- [37] OASIS eXtensible Access Control Markup Language (XACML) TC. <http://www.oasis-open.org/committees/xacml/>
- [38] A. Caltroni, V. Ciaschini, A. Ferraro, A. Ghiselli, G. Rubini and R. Zappi. G-PBox: A Policy Framework for Grid Environments. In *Proceedings of the International CHEP 2004*, 2004.
- [39] Cluster File Systems. Lustre: A Scalable, High-Performance File System. <http://www.lustre.org/docs/whitepaper.pdf>.
- [40] The GLUE Schema. GLUE Schema Specification - Version 1.2. <http://forge.ogf.org/sf/docman/do/listDocuments/projects.glue-wg/docman.root.drafts>.
- [41] Silicon Graphics Inc. (SGI) XFS: A high-performance journaling filesystem. <http://oss.sgi.com/projects/xfsl/>.
- [42] The EGRID project. <http://www.egrid.it/>