Integrated Authorization for Grid System Environments

Jiageng Li

Department of Computer Science, University of West Georgia,
1600 Maple St., Carrollton, GA 30118, USA

Abstract. Grid computing has received widespread attention in recent years as a significant new research field. Yet to date, there has been only a limited work on the grid system authorization problem. In this paper, we address the authorization problem and its requirements in a grid system environment. We propose a new integrated authorization service that tackles the authorization problem at two levels: grid system level and organization unit level. It is shown that the new approach not only meets the requirements of the authorization in grid system environment but also overcomes the disadvantages found in existing authorization designs.

1 Introduction

Grid computing has recently received significant attention as a new field in the area of distributed computing. When considering grid system security, the two principal issues are those of authentication and authorization. Authentication verifies the end user's identity, the first step in the authorization process. Authorization then determines if the (authenticated) user has the access rights on the requested resource or service. Specifically, when a server in a grid environment receives a client's service request, it asks two questions.

1. "Is the requesting client really the grid user that it claims to be?" (authentication, validating the client's identity)
2. "Does the grid user have the necessary permission to perform this service?" (authorization, checking client's rights)

The authorization problem in grid environments imposes some unique requirements on the system, due to its distinctive characteristics. The majority of existing research to date has taken place in the authentication arena. There has been only limited work on the grid system authorization problem. In this paper we propose a new integrated authorization service for grid system environments, focusing specifically on the issues associated with scalability. The remainder of the paper is organized as follows. In section 2, we investigate existing authorization designs in grid systems (e.g. Globus and Legion). In section 3, we present our integrated authorization service. We provide specifics regarding implementation of the service in section 4. In section 5, we conclude the paper.

Li J. (2005).
Integrated Authorization for Grid System Environments.
In *Proceedings of the 2nd International Workshop on Ubiquitous Computing*, pages 94-100
DOI: 10.5220/0002571300940100

2 Related Work

In this section, we critique the authorization system designs for two well-known grid systems: Globus [1] and Legion [2]. We observe that there are two main issues associated with the existing authorization designs in Globus and Legion. First, in both systems, the authorization policy is enforced at the local host level. When a client initiates a job, it has no apriori knowledge regarding its access rights to resources at other grid sites. As a result, the identification of resources that the user is authorized to use can be a trial-and-error process. A job request first has the resource management service identify potential system resources for use. However, whether the user is authorized to utilize these resources is not known until the job actually requests the resources. If authorization is denied, the process must be re-started from the beginning. Second, with the existing authorization mechanism, scaling of the maintenance and administration work is an issue. Authorization information is managed by each individual local site/object without a central control. Each site/object maintains its own authorization information for all its grid users. Information duplication, as well as administration and organization of this information in a coherent, consistent manner, becomes problematic.

An authorization design with a community authorization service (CAS) for grid systems is proposed in [3]. The basic idea is to divide the grid system into communities, with each community having its own CAS server. The CAS server manages the community security policy and makes access decisions for the community's resources. When a community member receives a resource request from a client, it sends the CAS server an authorization request. If the authorization request is consistent with its access control policy, the CAS server will generate a capability and send back to the community member. The resource allows the user access based upon this capability. In this approach, authorization is distributed yet limited to the scope of the community.

3 Integrated Authorization

Our integrated authorization service is based on a general grid system authorization model and authorization servers proposed in [4], within which the authorization takes place. We now present our authorization architecture for grid systems. The goal is to provide an approach that meets the requirements of a dynamic grid environment while overcoming the disadvantages identified in current approaches. The basic idea is to address authorization at two levels: the local organization unit level and grid system level. At the local organization unit level, each organization unit maintains its own authorization server that manages its authorization policies and makes access control decisions for the unit. At the grid system level, the organization units' authorization servers are linked together to form an integrated authorization service. This integrated service provides an authorization information query mechanism among multiple organization units at the grid system level.

3.1 Constructing an integrated authorization service

At the grid system level, authorization servers are linked together into a tree structure based on the grid system image. In addition to the authorization servers for each organization unit at the local level, an authorization server for the entire grid system is established for the grid, representing the root node of the tree. If an organization unit is further divided into multiple organization units, the authorization server of the original organization unit works as the parent node of its sub-divided organization units' authorization servers. In the authorization service, each authorization server is assigned a distinguished name that identifies it within the grid system. Each parent authorization server node contains the addresses of its direct subsidiary authorization servers which are represented as records in the parent node. In other words, in the tree-structured integrated authorization service, there are two types of authorization server node: leaf node, and non-leaf node. Only the leaf node contains the actual authorization information for the resources of its organization unit, while the non-leaf node only contains the pointers links to its direct subsidiary nodes.

On the other hand, each node except root node also keeps the address of its direct parent authorization server. If an organization unit is a direct branch of the grid system, then its authorization server is linked to the root authorization server as its parent node. In this way, an integrated authorization service is generated for the grid system. The service provides a flexible and powerful authorization information query system for use within multiple organization units or at the overall grid system scope.

Each authorization server maintains an authorization server database (ASD). An entry of the authorization server database is called an authorization server record (ASR). The record declares that a given organization unit is served by the authorization server. Each record consists of four fields: the organization unit name, record type, server type and the authorization server name/address. There are two types of records: the parent authorization server (PAS), and the children authorization servers (CAS). For the server type, the server can be either a primary server or a secondary server. In the integrated authorization service, an authorization server can have only one parent authorization server, while it can have multiple children authorization servers.

3.2 Authorization protocol

When the authorization service receives a client's authorization request, the request is routed to the specific authorization server that is capable of providing the appropriate authorization information. The routing procedure is motivated by Domain Name Service (DNS) [5].

Similar to DNS, we have two types of authorization query messages: recursive query and iterative query. In recursive query, the client asks the authorization server to provide a final answer for the query. In this case, if the server can solve the query, it checks its information base and responds. If the server can not solve the query, it sends the request to another server until it gets the answer or fails. On the other hand, the iterative query does not require as much work on queried server. In iterative query, if the authorization server can not solve the query, it will return the address of

an authorization server that it knows best to find the answer. In our approach, we use recursive query to query the local authorization server and use iterative query for the local authorization server to query other authorization servers. Furthermore, a local cache is used by each authorization server for recording the addresses of other authorization servers for future reference to expedite authorization queries.

During authorization, an authorization client first sends its request to the local authorization server of its own organization unit. If the authorization server can resolve the request, it will generate the result and send it back to its client. This is realized by checking OU information in the authorization request. If the OU name matches the authorization server's OU name in the request message, it means the authorization server is capable to resolve the query. If the authorization server can not resolve the query, it checks its cache first to see whether it has the address of the destination authorization server. If it is found successfully, it would send the query to the authorization server directly based on the address. On the other hand, if it is not found, it would send the request to its parent authorization server until either the request can be solved or it reaches the root authorization node. Upon the request reaching the root authorization server, the root server sends the authorization request to its child authorization server node according to the OU name in the authorization request until the authorization server is located. Then the authorization server will solve the query and send the authorization result back to the local authorization server. The local authorization server will correspondingly forward the authorization result to the original client. During the process, the local cache of the authorization server will record the addresses of the authorization servers it visited which are not available in the cache.

3.3 Message format

There are two types of query messages in the authorization process: recursive and iterative messages. Each query message includes two parts: header and individual queries. The header of query messages includes the Query_type, User_info, OU_name, and the Number of Queries. The Query_type is either "R" for recursive query or "I" for iterative query. User_info attribute is the global identity (GID) of the authorization client. OU_name represents the name of the remote organization unit with required resources. In one query message, it can contain multiple individual queries, but the multiple individual queries must be querying the same authorization server or organization unit. Each individual query includes Host_name, Resource_name, and requested Access_right for the remote authorization server. Three possible result messages for an authorization query are: Final query result, Referred query result with referred authorization server address, and Query failure. Each query result message also includes two parts: header and result part. The header of the result message includes the general information of the corresponding query message, which includes Code, Result_type/Failure_code, User_info, OU_name, and Number of Queries. The Code attribute in the header represents the query status. We use "0" to represent query success, and "1" query failure.

If the query status is a success, the Result_type attribute represents whether the result is the final query result or a referred query result with a referred authorization

server address. "*C*" represents the final query result and "*R*" represents a referred query result. A final query result can be a capability or a denied sign returned from the queried authorization server for each individual query. It is indicated by the ***Attr.*** field in the query result. If ***Attr***. field equals 0, it means a corresponding capability is returned; if ***Attr.*** field equals 1, it means the authorization request of the individual query is denied. A referred query result contains the referred authorization server address at the end of the result message.

Code (0): Result_type:
C: Final query result
Attr: (0) capability
(1) authorization denied
R: Referred query result
Referred authorization server address

If the query status is a failure, the attribute following the query status code is the failure code which indicates the failure reason of the query message. The failure can be authentication failure, server down, or query format error, etc. We use "*A*" to represent authentication failure; "*B*" for server unreachable/down; "*C*" for query format error and "*D*" for other errors.

Code (1): Failure_code:
A: Authentication failure
B: Server unreachable/down
C: Format error in query
D: Others

4 Implementing the Integration Authorization Service

In order to implement our authorization protocol, a number of implementation-specific issues must be addressed. In this section, the issues of service initialization, security, and failure handling for the integrated authorization service are each examined and resolved.

4.1 Initialization

When a new organization unit is available on the grid system, its authorization information should be added into the integrated authorization service of the grid system. The procedure can be divided into two phases: constructing a new authorization server and adding a new server into the system. During the first phase, the server establishes a secret key with each host housing shared resources within the organization unit that it manages. Note that the secret key is only shared between the host and its authorization server. Next, construct an authorization information base for the new authorization server. This information base stores the access control information of the organization unit. In order to do this it must poll its local grid users and hosts to identify shared resources. In the second phase, the new server is linked to its direct parent authorization server. The new authorization server determines which authori-

zation server should be its parent server; the position of the authorization server within the authorization service depends on the structure of the grid system. With the tree-structured authorization service, the authorization server needs to be registered and linked to its direct parent authorization server. To accomplish this, the system administrator of the new organization unit provides its direct parent authorization server with administrative and technical information about the organization unit and the authorization server. If the parent authorization server can acknowledge the request, it adds an entry for the new authorization server with the address, as a pointer link to it. The links are for constructing the integrated authorization service of the grid system, which are used for directing and routing authorization requests among authorization servers.

4.2 Security

Ensuring proper security for our integrated authorization service is vital. Several specific security facets must be addressed. First, the authorization service should only allow requests to take place between the authenticated clients and servers. Second, since the capability is the media tool for transferring the access permissions to the resources for the client, capabilities must be transmitted securely and can not be compromised by a third party. Third, capabilities must not be generated by anyone other than the authorization servers. Fourth, the authorization server must not be impersonated by another entity on the network. In our integrated authorization service, the security issues are addressed both at the grid system level and at each local authorization server:

- At the grid system level, with the public key infrastructure, each authorization server and authorization client has a pair of keys: a public key and a private key. The combination of public key infrastructure and SSL is used to ensure the security of the communication between the authorization client and authorization server, and between authorization servers.
- On the other hand, each authorization server and the hosts with shared resources within the organization unit scope shares a secret key at the local level. Capabilities are generated by encrypting the authorization information with the corresponding secret key. With the secret key, it is assured that the capability can only be decrypted and interpreted by the two parties. The subject of the user's public key, as part of the authorization information, is encrypted within the capability. Therefore, the capability can only be interpreted by the resource holder and can only be used by the user with the correct public key.

4.3 Failure handling and recovery

Since the authorization server is responsible for managing the authorization functions for an organization unit, if the authorization server fails, it would cause the resources of the organization unit beyond access and sharing.

For the failure handling and recovery, in our authorization service, we assume that each authorization server is backed up with at least one secondary server. The primary and secondary authorization server(s) should have the same authorization information and be kept synchronized. Updates are performed on the primary server first. When such an update takes place, the primary server sends a synchronization message to its secondary server(s) to keep their authorization information synchronized.

We assume that in the integrated authorization service, the parent authorization server has the knowledge of the addresses of all the authorization servers, primary or secondary, of its subsidiary organization units. Each subsidiary authorization server also knows all the addresses of its direct parent authorization servers. The authorization request is always sent to primary authorization server first. When a client queries a failed primary authorization server, after a time-out, the query will be resent to its secondary authorization server. Finally, it is noted that secondary servers can be used to split the workload during times of activity.

5 Conclusion

In this paper, we presented a new integrated authorization approach in grid system environments. We addressed the problem of existing methods at two levels: the grid system level and the local organization unit level. By dividing the grid system into organization units, a centralized authorization server for each of them is set up at the local organization unit level. On the other hand, at the grid system level, the authorization servers are interconnected into a hierarchical tree structure to generate an integrated authorization service. With the new approach, scalability is solved with the independent, distributed servers. As the grid system expands, additional authorization servers for the new organization units can be added into the integrated authorization service without much overhead. This new authorization approach also addresses the concerns of existing grid system authorization approaches.

References

1. Foster, I., Kesselman, C.: The Globus Project: A Status Report. Proceedings of IPPS/SPDP'98 Heterogeneous Computing Workshop, (1998) 4-18.
2. Grimshaw, A., Wulf, W.A.: the Legion team. The Legion Vision of a Worldwide Virtual Computer. Communications of the ACM, Vol. 40, No 1, (1997) 39-45.
3. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A Community Authorization Service for Group Collaboration. Proceedings of the Third IEEE International Workshop on Policies for Distributed Systems and Networks, (2002) 50-59.
4. Li, J., Cordes, D.: Authorization in Grid System Environments. Proceedings of the 41st ACM Southeast Regional Conference, (2003) 292-297.
5. RFC1034: Domain Names, Concepts and Facilities.