Dynamic and Fine-Grained Authentication and Authorization Architecture for Grid Computing

Hyunjoon Jung, Hyuck Han, Hyungsoo Jung, and Heon Y. Yeom

School of Computer Science and Engineering, Seoul National University,
Seoul, 151-744, South Korea
{hjjung, hhyuck, jhs, yeom}@dcslab.snu.ac.kr

Abstract. The Globus Toolkit makes it very easy and comfortable for grid users to develop and deploy grid service. As for the security mechanism, however, only static authentication and coarse-grained authorization mechanism is provided in current Globus Toolkit. In this paper we address the limitations of current security mechanism in the Globus Toolkit and propose a new architecture which provides fine-grained and flexible security mechanism. To implement this without modifying existing components, we make use of the Aspect-Oriented Programming technique.

1 Introduction

With the advent of grid computing, Globus Toolkit[8] have been playing an important role for making grid systems. Thanks to the support of various components in the Globus Toolkit, many grid developers have been able to develop their own grid systems very conveniently. When the grid service administrator - in this paper, we would like to divide grid users into grid service administrator and general grid users for explaining more precisely - make their own grid systems, the grid computing environment usually consists of many service nodes which are managed correlatively. Namely, many nodes can be separated logically into specific domains which can be under the same policy, so called virtual organization [11]. The grid service administrator should manage various kinds of services according to the policy of each domain.

Although it is possible in current Globus Toolkit, there are some limitations. If a new user wants to use some grid service, the grid service administrator must change the policy of the service which might be currently running. However, to change the security policy of a specific grid service, it is required to stop the grid services related to the altered security policy. After the policy change, the grid services are redeployed according to the new security policy.

Besides, current security scheme of Globus Toolkit supports only service level authorization. In other words, method level authorization is not supported. If the grid service administrator wants to configure different authorization level for each method, he should make extra grid service under different security policy. Consequently, current security configuration scheme in Globus Toolkit is static and coarse-grained, which results in extra jobs for grid service administrator.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3516, pp. 179–186, 2005.

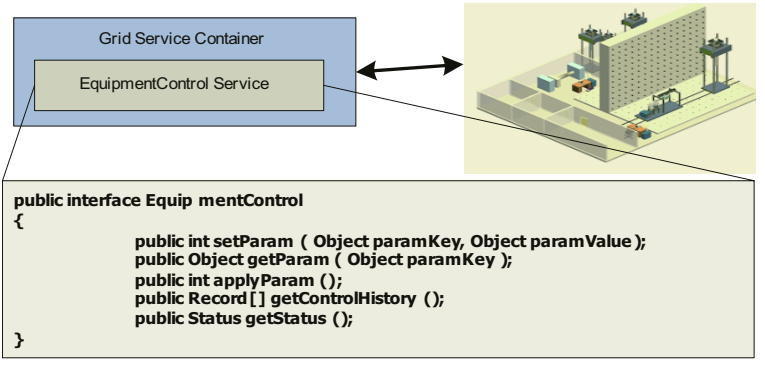

Fig. 1. Motivation Example

To mitigate these limitations, we present a dynamic authentication and authorization scheme. Our scheme also allows fine-grained method level security scheme.In designing and implementating this scheme, we have used *Aspect-Oriented Programming* [9, 10, 14] so that existing Globus Toolkit is left unmodified.

The rest of this paper is organized as follows. We discuss the motivation of this project in detail in section 2 and introduce our design goals and strategies in section 3. And then we propose new architecture in section 4. In section 5, we explain how to implement our scheme with *Aspect-Oriented Programming*. The section 6 addresses issues of grid security related to our work and future works. In the end, we conclude our paper with summary and contribution.

2 Motivation

In this section, we describe the limitations of security scheme in current Globus Toolkit. And then our focuses of this paper will be introduced.

In grid computing, there are many users who try to access some services. And there are also many kinds of grid services being executed by user's request. To manage users and services securely, grid computing have adopted security policies for each user and service. In practice, Globus Toolkit ensures the security of grid service and user with exploiting grid security infrastructure (GSI) [12].

With the advent of web-based architecture, grid service also have evolved into conforming architecture[5]. However, current Globus Toolkit supports only static management of security policy in web-based architecture. In other words, the security policy of specific service would be set before deploying. Besides, while that service is running, security policy cannot be reconfigured and applied instantly to that service. Owing to this kinds of limitations, grid service administrator should first stop the service, reconfigure specific security policy and then redeploy that service. Since this reconfiguration and redeployment should be done for each small change, the burden on the part of the administrator could be huge.

In addition to the static management of security policy, security level in authorization can be also considered as a limitation. When the grid service administrator deploys some services with current Globus Toolkit, the level of authorization is the service instance. Figure 1 describes the motivating example. When *EquipmentControl* service is deployed, grid administrators configure security policies for each user level. If *EquipmentControl* grid service would be subject to the policy of table1, it is impossible to provide service to each user's permission using existing Globus Toolkit.

Table 1. The list of user's permission of each service method

user group	defined security policy
experiment, operator	permit all methods
other reseacher	permit getXXX() methods
guest	permit getControlHistory() method

In other words, current Globus Toolkit cannot support method-level authorization scheme. Since only service-level authorization scheme is provided, the grid service administrator ought to make each service separately according to each user's request and authorization policy. Consequently, it leads to the inefficient utilization of resources.

3 Design Goals and Strategy

As stated in the previous section, our goal in this paper is to ameliorate the current Globus Toolkit as follows:

- To provide dynamic authentication and authorization
- To support fine-grained authorization
- To implement this scheme as an extra module without modifying the Globus Toolkit.

To provide dynamic authentication and authorization means that grid service administrators would be able to reconfigure the security policy of running services without suspending them. Supporting fine-grained authorization presents enhancements of security granularity and removes the redundancy of grid services. For instance, if some grid administrator would like to change security policy of some grid services, they would be troubled like the example of the section2. It is an inefficient utilization of resources due to multiple security policies. Therefore, method level authorization with dynamicity would be a better alternative to solve this kind of problem. In order to support these two security schemes in the Globus Toolkit, it would be easy to implement if we modify security part in Globus Toolkit.

In that case, however, Globus Toolkit should be recompiled and then it causes undesirable extra needs. For these challenges of implementation, we adopt gracious programming skill that is *Aspect-Oriented Programming*. It ensures that

any existing program does not need to be modified. By exploiting this way, we would present more compact and modularized architecture. Aspect-oriented programming would be explained down to the minutest details in the implementation part.

4 Proposed Architecture

To achieve our goals, we propose the architecture composed of three components in accordance with each role as shown in figure2.

4.1 FSecurity Manager

Originally, security mechanism of Globus Toolkit is mainly divided into authentication part which verifies someone and authorization part which checks the right of a user in grid service. And the policy of the authentication and authorization can be set only ahead of the deployment of grid service. Hence, to achieve the dynamicity of security scheme, *FSecurity Manager* intercepts the request of authentication and authorization in current Globus Toolkit when the request is invoked. And then, that request is processed in *FSecurity Manager* with Security Configuration Data.

For instance, we assume that the grid administrator expects 10 users would use his grid service A before deployment. But, after some time, 5 new users would want to use the grid service A. In that case, the grid administrator must stop the grid service A to modify new security policy for the 5 new users. And after applying new security policy, grid service will be executed again correctly. However, this kind of process would be changed with *FSecurity Manager* as follows. Above all, the most troubled part which do not check the security policy at running time is changed with intercepting request of *FSecurity Manager*.

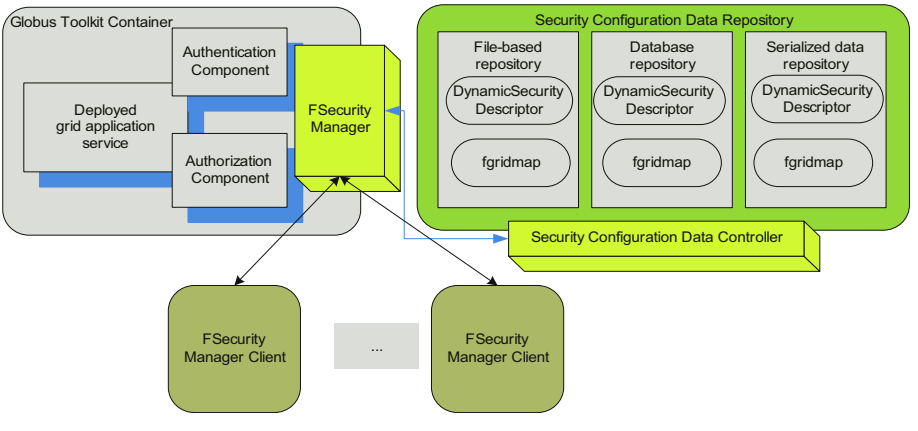

Fig. 2. Proposed architecture

And then *FSecurity Manager* reads the security policy at that time, applied that scheme to authentication and authorization instantly. Therefore, *FSecurity Manager* supports the reconfigurable security processing.

Moreover, *FSecurity Manager* provides a fine-grained authorization processing with adding the element which describes the relation between user and grid service method. In current Globus Toolkit, before the service deployment, grid service administrator used to configure the gridmap file which describes simply those who can use specific grid service. However, that mechanism is changed into supporting the dynamic management and the fine-grained description with *FSecurity Manager*. In the former case, grid service administrator can reconfigure gridmap file even though service is executing. There is no need to stop the service and redeploy it later. And as for the latter, the description which some user have rights to use some grid service method is added into new type gridmap file, which is called fgridmap.

4.2 Security Configuration Data Controller

Security Configuration Data Controller is an extra module in order to manage security policy information. Security policy information can be stored as a XML file, serialized file or database records. To support this kind of function, this controller is responsible for connecting, storing, and loading specific security policy information with a data repository.In real grid computing, grid administrator can install this part in a machine which Globus Toolkit is executed or another machine which may be dedicated to data repository.

4.3 FSecurity Manager Client

This component supports that grid service administrator controls and manages the security policy of grid service from a remote or local node. Client program can be installed in a ordinary computing node or mobile node such as PDA, mobile phone. Also the grid service administrator can manage a *FSecurity Manager* via web interface.

5 Implementation

As we mentioned above, our implementation without the change of Globus Tookits is done with a refined programming skill which is Aspect-Oriented Programming. In this part, we would describe the reason why we have adopt Aspect-Oriented Programming and how it was applied. And then the operation flow of the proposed architecture would be explained with a structural perspective.

5.1 Aspect-Oriented Programming and Our Architecture

Aspect-Oriented Programming (AOP) complements *Object-Oriented programming* by allowing the developer to dynamically modify the static OO model to create a system that can grow to meet new requirements. The reason why we use *Aspect-Oriented Programming* technology is to separate the core part of

program from the extra part of program because there is some kinds of different properties between the former and the latter. For instance, many of us have developed simple web applications that use servlets as the entry point, where a servlet accepts the values of a HTML form, binds them to an object, passes them into the application to be processed, and then returns a response to the user.The core part of the servlet may be very simple, with only the minimum amount of code required to fulfill the use-case being modeled. The code, however, often inflates to three to four times its original size by the time extra requirements such as exception handling, security, and logging have been implemented.

Therefore, AOP allows us to dynamically modify our static model to include the code required to satisfy the extra requirements without having to modify the original static model - in fact, we don't even need to have the original code. Better still, we can often keep this additional code in a single location rather than having to scatter it across the existing model, as we would have to if we were using OO on its own.

The Globus Toolkit 3 is also implemented with Object-Oriented programming. Therefore, the supplement of Globus Toolkit should conform to the Object Oriented model. However, Globus Toolkit is so complicated and complex. And the security part of Globus Toolkit are involved with almost area. On the ground of that, this case would be modeled as the cross-cutting concerns which are named in aspect-oriented programming.

Whenever grid users try to access some grid services, security component must be activated and then perform the process of authentication and authorization. Hence, the invocation point of security component in Globus Toolkit should be defined as point-cut in aspect-oriented programming which is the term given to the point of execution in the application at which cross-cutting concern needs to be applied. And then to perform some predefined action, code will be implemented as an advice which is the additional code that someone wants to apply to the existing model. As a result, aspect-oriented programming is exploited to instantiate proposed architecture using point-cut and advice which is named in that field. Besides, it is possible to implement the proposed architecture without any modification of Globus Toolkit thanks to this programming technique.

5.2 Operational Flow

In this section, we describe the implementation details of the proposed architecture. The implementation was done using AspectJ[13].

Let us first look at the left side of figure3. It shows how the authentication is processed in our architecture. Originally, initSecurityDescriptor and getSecurityDescriptor are performed in current Globus Toolkit. However, there's only one time check which means getSecurityDescriptor does not check SecurityDescriptor at running time. Hence, we defined two point-cuts at the invocation time of initSecurityDescriptor and getSecutityDescriptor. In this way, any invocation of SecurityDescriptor can be substituted DynamicSecurityConfig which always

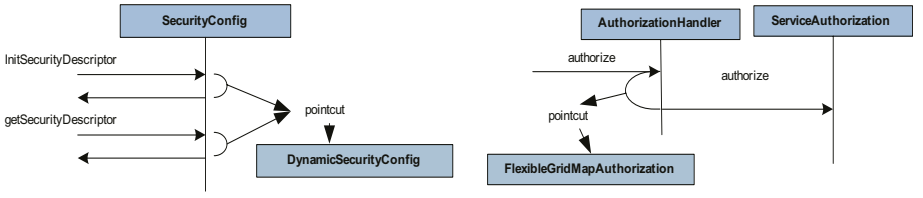

Fig. 3. The operational flow of authentication part(left) and authorization part(right)

checks the SecurityDescriptor and applies up-to-the-minute information to the specific grid service at running time.

In authorization, FlexibleGridMap Authorization (fgridmap authorization) functions as the core. Whenever AuthorizationHandler is invocated, the pointcut leads the execution flow to fgridmap authorization. And then operation would be continued with the authorization which conform the defined policy between the user and grid service method. The style of fgridmap is illustrated in the right side of figure3. This authorization part can be performed using the latest authorization information for the specific grid service no matter when the update has been made. As was done in authentication part, the processing of authorization is performed at running time with the help of aspect-oriented programming.

To explain the whole flow of operation with authentication and authorization, let us assume that there are grid user 1, grid user 2, and grid user 3. Also, there is a grid service S1 which can provide methodA, methodB, and methodC. If user 1 who is set to use methodA and methodB, the request for methodC by user 1 shuld be denied by the authorization part. Even if new grid user 4 who is not set on DynamicSecurityDescriptor at deployment time, grid user 4 may be added according to the decision of grid service administrator. And then the request by grid user 4 in compliance with the DynamicSecurityDescriptor and the FlexibleGridmap may be accepted or denied.

To sum up, with the proposed architecture, grid service administrator can configure the authorization scheme up to method level. Moreover, grid service administrator can reconfigure the authentication and authorization policy at service running time. It enhances the limitation of Globus Toolkit by using aspect-oriented programming.

6 Conclusion

In this paper, we present a dynamic and fine-grained security architecture for grid service administrators who use Globus Toolkit. With this component, the extra needs of many grid administrators who frequently have to reconfigure security policies and redeploy grid services can be alleviated very simply.

To make current Globus Toolkit more efficient, we have studied the limitation of security part of current Globus Toolkit and then proposed some refinements of this research. The proposed architecture has been implemented using Aspect-

Oriented Programming technique without modyfing Globus Toolkit. Therefore, in real grid computing, grid service administrator can use this component very conveniently with existing Globus Toolkit.

References

1. Von Welchr, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder and Frank Siebenlist "X.509 Proxy Certificates for Dynamic Delegation", In Annual PKI R&D workshop, April,2004
2. Von Welch, Frank Siebenlis, Ian Foster, John Bresnaban, Karl Czajkowski, Jarek Gawor, Carl Kesselman, Sam Meder, Laura Pearlman and Steven Tuecke , "Security for Grid Services", In IEEE Symposium on High Performance and Distributed Computing ,June,2003.
3. Ian Foster, Carl Kesselman, Cene Tsudik and Steven Tuecke "A Security Architecture for Computational Grids", In 5th ACM Conference on Computer and Communication Security, 1998.
4. "Security in a Web Services World: A Proposed Architecture and Roadmap", A Joint White Paper from IBM Corporation and Microsoft Corporation, April 2002.
5. Tuecke S. and Czajkowski K. and Foster I. and Frey J and Graham S. and Kesselman C. and Maguire T. and Sandholm T. and Snelling D. and Vanderbilt P "Open Grid Services Infrastructure(OGSI) Version 1.0", Global Grid Forum, June 2003.
6. Nataraj Nagaratnam, Philippe Janson, John Dayka, Anthony Nadalin, Frank Siebenlist, Von Welch, Ian Foster and Steve Tuecke "The Security Architecture for Open Grid Services", July 2002.
7. Foster, I., Kesselman, C., Nick, J. and Tuecke, S. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration " , Globus Project, 2002. http://www.globus.org/research/papers/ogsa.pdf.
8. Jarek Gawor, Sam Meder, Frank Siebenlist and Von Welch, "GT3 Grid Security Infrastructure Overview", In ,Globus Project 2003.
9. the AspectJ Team, "The AspectJTM Programming Guide", Copyright (c) 1998-2001 Xerox Corporation, 2002-2003 Palo Alto Research Center, Incorporated. All rights reserved.
10. Markus Voelter, "Aspectj-Oriented Programming in Java" in the January 2000 issue of the Java Report
11. Foster, I. and Kesselman, C. and Tuecke, S., "The anatomy of the grid : Enabling scalable virtual organizations", Intl. J. Supercomputer Applications, 2001
12. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke., "Security Architecture for Computational Grids", 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.
13. Palo Alto Research Center, "The AspectJ(TM) Programming Guide", http://eclipse.org/aspectj/
14. Tzilla Elrad ,Robert E. Filman and Atef Bader "Aspect-oriented programming: Introduction", In Communications of the ACM , 2001