Policy Management for OGSA Applications as Grid Services (Work in Progress)

Lavanya Ramakrishnan
MCNC-RDI Research and Development Institute
3021 Cornwallis Road, P.O. Box 13910, Research Triangle Park, NC 27709-2889
lavanya@cnidr.org

Abstract

We present here two Grid services - PolicyManagerService and AuthorizationService. These two OGSA security services aid application grid services to manage policy and enforce policy decisions in a service domain. A simple policy decision point using XACML is used to make a policy decision that is returned to the enforcement point. Using the OGSA notification mechanism the policy is synchronized across the two services. The paper discusses the architecture and the early implementation using the Globus toolkit 3.0 and Sun's XACML implementation.

1. Introduction

GridIR is a system based on the Open Grid Service Architecture (OGSA) [1] framework, enabling complex information retrieval on the grid. The GridIR system consists of the following services: Collection Management Services - to control collection, harvesting of data; Indexing and Searching Services - to build indices from document collections; and Query Processing Services - for distributed searching and results merging. The GridIR services have various security requirements. They need to manage credentials of the services and the users of the system. This could be complicated as there maybe persistent queries that the service may be responsible to use the user's delegated credential for long after the user has disconnected from the service. In addition access to the data and the operations supported by the service will need to be restricted based on some policy. At the same time it is desired that the security mechanisms be independent of the application logic.

Thus we see that grid services require a security infrastructure to support its activities. The OGSA Security Architecture [2] talks about how some of the security functionality could be composed as grid services. This paper addresses some of the authorization requirements of grid applications such as GridIR composed as OGSA services and looks at a possible architecture to mange policy and trust in such an environment. This work is a preliminary design and results from a prototype implementation based on the current version of the OGSA architecture and OGSI [3] specifications as implemented in the Globus toolkit 3.0. This work serves as an - 1) Illustration of a use-case and early implementation results of OGSA services. 2) Realization of some of the security support required by OGSA services 3) Basis for discussion on standarization of interfaces and data representation for policy management.

We introduce in this paper [1] two OGSA grid services that applications could use to manage policy and trust issues and enforce authorization checks. The PolicyManagerService is responsible for policy representation and managing policy data during the lifetime of the services. The AuthorizationService aids the application service enforce the policy when clients use their services. The material presented in this paper is part of a larger architecture for providing a security infrastructure for OGSA grid services such as GridIR.

2. Architecture

We present here a scalable, flexible security infrastructure used by grid services for policy management in the OGSA architecture. In Figure 1, we illustrate how the PolicyManagerService and AuthorizationService would interact with each other to setup an environment for policy management for the service. In this architecture we see that an AuthorizationService could be created at the application service creation time or could use an existing service. The AuthorizationService also subscribes to notifications from the PolicyManagerService on changes to policy of this service instance.

[1]The material is based upon work supported by NASA under award No(s) NAG 2-1467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

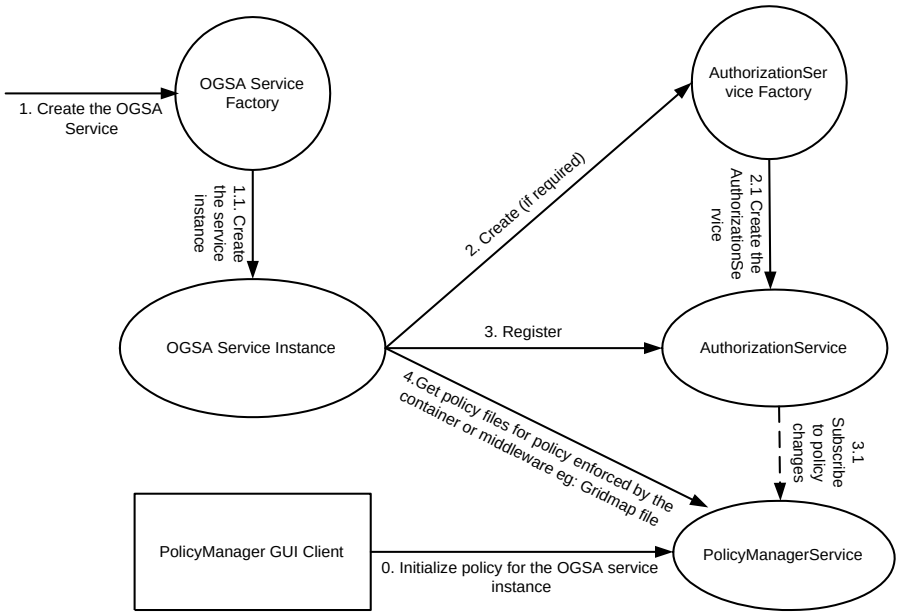

Figure 1. PolicyManagerService and AuthorizationService interaction during service creation time

In Figure 2 ,we illustrate how the AuthorizationService can be used by the OGSA service to make policy decisions on its behalf. The AuthorizationService would get the policy files from the PolicyManagerService, if required and use the internal XACML based Policy Decision Point (PDP) to make the decision. It is anticipated that the AuthorizationService would have a cache of the policy files that it would update only when it recieves notifications of changes to the policy associated with the service. Thus it is anticipated that most decisions would be served from the local cache avoiding a call to the PolicyManagerService each time.

2.1. Separation of duty

The functionality between the policy management and the policy decision point has been separated in two separate services - PolicyManagementService and AuthorizationService. The PolicyManagementService is responsible for creating, managing and updating the policy information of the services. The AuthorizationService acts as a simple Policy Decision Point that uses the policy information from one or more PolicyManagementService to decide whether a client should be allowed to access the requested resource.

The policy associated with a service instance is added to the policy repository through the PolicyManagerService API. The policy can be updated dynamically through the PolicyManagerService interface. A service instance may create its own personal AuthorizationService during the post-creation process. It also registers with an AuthorizationService giving it the GSH of the PolicyManagerService that holds the policy files for the service instance.

While enforcing the various rings of security on a grid service, it is likely that the access rules over the layers may need to be synchronized. For instance in our prototype implementation, the Globus toolkit 3.0 [4] enforces the service-level authorization through gridmap files. The GridIR application needed the ability to express and enforce policy at a much finer level of granularity, we implemented the the PolicyManagerService to generate both the gridmap file and the XACML files required by the AuthorizationService. The service instance obtains the gridmap file from the PolicyManagerService and configures the instance-gridmap parameter during the post-creation process of the service instance.

2.2. Dynamic Policy

Trust relationships continually change during the lifetime of the service instance. Hence it is required to have the ability to dynamically update policy information. The policy information can be updated at any point through the PolicyManagerService interface. The AuthorizationService can subscribe to the PolicyManagerService for updates on changes in the policy. This allows the policy to be updated dynamically throughout the system.

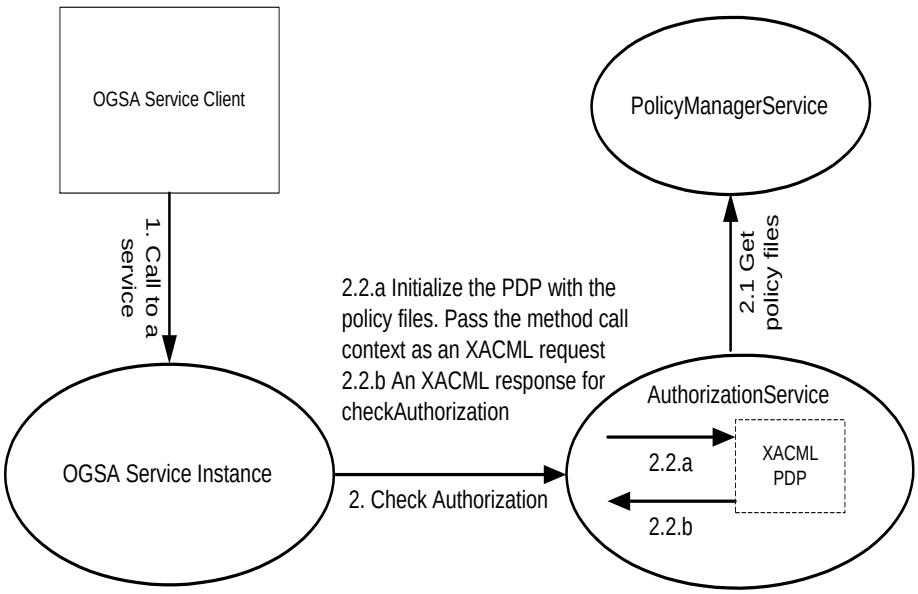

Figure 2. PolicyManagerService and AuthorizationService interaction during service call time

2.3. Scalability

The separation of functionality between the policy management and decision point makes the architecture scalable. Multiple AuthorizationServices and PolicyManagerServices can be deployed and connected in a hierarchical model to ensure scalability of these functionalities

2.4. Pluggability

The architecture illustrated allows for pluggability of various components. It lets the user use various implementations of the AuthorizationService and PolicyManagerService as may be required by applications. This allows us to enforce application specific security while still keeping the security mechanisms independent of the application logic.

2.5. Usability

XACML is used as a policy language to express the complex rules to be expressed. It is often tedious and cumbersome to write these policy files directly in XML format. Thus to reduce the burden on the end-user we provide a graphical user interface that can be used to enter the policy rules for the service. The graphical user interface module parses the GWSDL of the service and generates a list of entities on which policy may be specified. In the prototype implementation it parses the GWSDL and shows the service name with its associated portTypes and corresponding methods. The user thus can choose to specify the policy at one or more of service, portType and method levels.

The user interface then acts as a client to the PolicyManagerService that processes the information to generate the corresponding gridmap and XACML policy files. These policy files are associated with the Grid Service Handle (GSH) of the service. This helps to uniquely identify the policies associated with a service. An AuthorizationService uses the GSH of a service to fetch the policy files from the PolicyManagerService.

2.6. Flexibility

This paper illustrates how the PolicyManagerService and AuthorizationService can be used to secure the services. We see that the same infrastructure can be used to enforce policy decisions at various levels. In the prototype implementation access to the operations offered by the services is restricted based on policy-decision made by the AuthorizationService. The PolicyManagerService and AuthorizationService could be used to restrict access to service-data of the service. Thus the flexibility of where and when to make policy decisions is left to the application while not burdening the application with the actual security logic.

2.7. Trust between entities

We see here that the PolicyManagementService and AuthorizationService are OGSA security services. It is ex-

tremely important that the access to these services is also restricted based on some policy. The security services could enforce policy in a similar manner to other OGSA services as illustrated in Figures 1 and 2 or may use some other internal libraries to do the same. By separating the security services from the application logic, exposure of the functionality to a large-scale of end-users has been limited.

The OGSA application service, the AuthorizationService and PolicyManagerService form a triangle of trust. The PolicyManagerService would restrict access to the methods to create and change policy to the service-owners and only trusted AuthorizationServices would be allowed to get the policy from the PolicyManagerService. The PolicyManagerService would also need to enforce that a particular AuthorizationService can see only the policies of the services that it is responsible for. It is also anticipated that for highly sensitive applications and/or in production environments the AuthorizationService and PolicyManagerService would be run on highly secure machines probably at the level of a Kerberos server.

3. Discussions

3.1. Scenarios

The OGSA architecture allows the use of OGSA services in various contexts and combinations. Here we present some discussions on the possible use-cases of the PolicyManagerService and AuthorizationService may be used with the application grid services.

3.1.1 Personal Policy Manager and Authorization Service

An application with a larger number of users or a specialized policy mechanism may choose to have associated with it an instance of, each of, PolicyManagerService and AuthorizationService. This can be considered as a personal mode where the security services would have the same lifetime as the application service instance. There is therefore a 1:1 relation between the application service and security services.

3.1.2 Group Policy Manager and Authorization Service

In most cases we anticipate that a group of application services would use one PolicyManagerService and AuthorizationService. It is likely the services would be grouped by a specific feature such as the level of security they would like to enforce. Thus the ServiceGroup idea from the OGSI specification could be used to associate the PolicyManagerService and AuthorizationService with a ServiceGroup.

3.1.3 Multiple Policy Manager and Authorization Services

It is also likely that there may be multiple PolicyManagerServices and AuthorizationServices in a service domain that enforce various kinds of policy rules. And in a lot of cases there may be services that may want to use a set of different policy conditions for enforcing authorization decisions. For example: an OGSA service may allow access to a certain resource based on some role-based-access control and in addition may want to check the number of resources the user is currently using before granting access. Such policies and authorization decisions would be typically available from different services. Thus there may be multiple Authorization and Policy Manager services that may be connected together before a final decision is computed and enforced.

3.2. ServiceData and Notifications

We use the service data and notification mechanisms available with OGSI to implement synchronization between the PolicyManagerService and AuthorizationService. The PolicyManagerService maintains service data that is used to reflect any changes made to the policy to all its subscribers. When an application service registers itself with the AuthorizationService, the service subscribes to notifications on policy updates of this service. The first time a checkAuthorization call is made, the AuthorizationService would fetch the policy files from the PolicyManagerService and maintain it locally. Thereafter for all policy checks the local copy of the policy will be used. When there is a policy update, the PolicyManagerService notifies the AuthorizationService of the change. The AuthorizationService then invalidates its local copy. Thus on the next call to checkAuthorization a fresh copy of the policy will be obtained. The performance of the system is thus improved since the AuthorizationService does not need to get the policy files from the PolicyManager each time.

3.3. PolicyRepresentation

The policy used by the security services is constructed using the XACML syntax (Figure **??**). The same policy syntax can be used to specify policy for various elements of the system that need to have restricted access. We use the Grid Service Handle (GSH) of the service instance as the ResourceId in the policy. This helps us uniquely identify the service instances across the entire service domain. The notion of sameness of service instances as mentioned in the OGSI specifications holds here. If an implementation of a service instance is distributed or replicated across many machines for scalability, all of them may be identified by the same GSH. In this case, it would be desired that the policy

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="Policy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm
:first-applicable">
<Target>
<Subjects><AnySubject/></Subjects>
<Resources><Resource><ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">GSH of the service</AttributeValue>
<ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oa
sis:names:tc:xacml:1.0:resource:resource-id"/>
</ResourceMatch></Resource></Resources>
<Actions><AnyAction/></Actions>
</Target>

<Rule RuleId="AccessRuleForOperationName" Effect="Permit">
<Target><Subjects><AnySubject/></Subjects>
<Resources><AnyResource/></Resources>
<Actions><Action>
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">OperationName</AttributeValue>
<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="MethodNa
me"/>
</ActionMatch>
</Action></Actions>
</Target>

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Distinguished Name of trusted client</AttributeValue>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
<EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="DN"
/></Apply></Apply>
</Condition>

</Rule>
<Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

Figure 3. Sample Policy

associated with these instances are the same, maintaining the uniqueness property. Thus we see that the uniqueness of the GSH for policy representation would hold in this situation.

After the intial header section identifying the service, we see that access rules for each of the elements to be restricted can be specified. The OperationName in case of method access or ServiceDataId incase of service data can be specified in the Action section. With each action a set of identities is associated that is the set of authorized users allowed to perform that action.

4. Implementation

The prototype implementation of the PolicyManagerService and AuthorizationService is implemented using the Globus toolkit 3.0. We use SUN's open-source XACML implementation to implement the Policy Decision Point.

4.1. Interfaces

The current interfaces of the PolicyManagerService and AuthorizationService are shown in the tables. We anticipate that in the future one or more of these interfaces are likely to get standardized. They are also likely to change depending on the needs of the applications.

4.2. Policy Representation

We use XML Access Control Markup Language (XACML) for representing the policy associated with the services. XACML is an XML based policy language or schema designed to be able to specify policies to use to control access to applications. The access-control policy language of XACML lets users specify the rules about who can do what and when.

The policy entered at service and portType levels gets translated to equivalent method level in this early implementation. The policy is represented as a list of Distinguished Names from X.509 certificates that are allowed to access the method. It is expected that in later versions other features of XACML like the ability to specify time constraints on the policy will also be used. The GSH of the service is used as a unique identifier to associate a service-instance to a policy.

4.3. Policy Decision Point

XACML also provides a request/response language. The request schema is used by applications to present an XML request for access. The corresponding response for the request is also defined as part of the XACML language. At the Policy Decision Point ie the AuthorizationService the request for checking authorization is translated into the XACML request format and passed onto an embedded XACML Decision Point alongwith the associated policy files. The result of the response in XACML language is translated as a boolean representing whether the request access should be granted or denied.

5. Future Directions

We would like to extend the policy representation and management interface to be able to specify other constraints like time conditions.

Performance measurements of the calls to the PolicyManagerService and the AuthorizationService will be required to identify how the system could be improved.

We would like to experiment with various combinations of running the AuthorizationService and the PolicyManagerService to understand additional issues that may arise while linking multiple instances of these services.

6. Conclusions

This paper shows how the PolicyManagerService and AuthorizationService could be used to manage and enforce policy decisions for OGSA services. Separation of duty between the policy management and decision points helps

Table 1. PolicyManagerService interface

Operation Name	Input Message	Output Message
generatePolicy	serviceId - xsd:string, Representation of service WSDL in a custom data structure, Representation of entered ACL in custom data structure	Success - xsd:boolean
updatePolicy	serviceId - xsd:string, Representation of service WSDL in a custom data structure, Representation of entered ACL in custom data structure	Success - xsd:boolean
getGridmap	serviceId xsd:String	gridmapFilePath xsd:string
getACL	serviceId xsd:String	policyFilePath[ ] xsd:string

Table 2. AuthorizationService interface

Operation Name	Input Message	Output Message
register	policyManagerHandle - xsd:string	-
checkAuthorization	Representation of context containing client's Id, service and operation being accessed	authorizedValue - xsd:boolean

build a flexible and scalable architecture. With the help of notification of service-data the policy updates are transferred throughout the system.

The policy can be enforced at various points in the OGSA application services - at portType, the method or service-data level, etc. Using the XACML policy schema the policy is represented to say who (DN from an X.509 certificates) can access what (operation or action).

7. Acknowledgements

The author would like to thank Sousan Karimi, Kevin Gamiel, Jeremiah Morris and Travis Walsh for discussions on the security requirements and testing out early versions of the services.

References

[1] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration. *Open Grid Services Architecture WG, Global Grid Forum*, 2.9(Draft), June 2002.

[2] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, and S. Tuecke. The Security Architecture for Open Grid Services. *Open Grid Services Security Architecture WG, Global Grid Forum*, 2.9(Draft Version 1), July 2002.

[3] S. Tueke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open Grid Services Infrastructure(OGSI) Version 1.0. *Global Grid Forum GFD-R-P.15(Proposed Recommendation)*.

[4] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, and S. Meder. Security for Grid Services. *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press*, June 2003.