GridShib and PERMIS Integration

Andrey Novikov (an64@kent.ac.uk)
GridShib and PERMIS Integration

Andrey Novikov
Computing Laboratory, University of Kent
an64@kent.ac.uk

Abstract

This paper describes the results of our recent GridShibPERMIS project to provide policy driven role-based access control decision making to Grid jobs, in which the user’s attributes are provided by a Shibboleth Identity Provider (IdP). The goal of the project is to integrate the identity federation and attribute assignment functions of Shibboleth with the policy-based enforcement function offered by the PERMIS access control infrastructure to authorize Grid jobs running with Globus Toolkit v4. This was done by studying the GridShib project, the existing integration of Globus Toolkit and Shibboleth and developing the GridShibPERMIS Context Handler which allows for interoperability between GridShib and PERMIS software by providing the required attribute extraction, conversion and transfer functions. As a result, the GridShibPERMIS project integrates the advantages of both Shibboleth cross-organization identity federation and PERMIS policy driven role-based access controls and represents a new avenue of policy-based authorization for Grids. This project will provide an efficient and simple mechanism, having low deployment costs, for enforcing Grid security. The paper provides a brief overview of the technologies involved, GT4, Shibboleth and PERMIS, and presents the work done, as well as the discussions of lessons learned and our future plans.

1. Introduction

Current progress in Grid computing [8] opens up new opportunities for the distributed multi-institutional collaborations forming Virtual Organisations (VOs) [7]. It dramatically increases the ability of academic institutions to accumulate and analyze data for research purposes, but in turn raises issues of secure authentication and authorization. At the moment, in the existing Grid solutions, such as the Globus Toolkit [9], they are usually based on the identities of interacting parties. However, with the expansion of VOs this approach seems to lack scalability and flexibility, as it is unable to cope with dynamically changing rights and permissions, as well as raises privacy concerns due to the release of personal information for user authorization.

The outlined problems can be solved by expressing users’ rights as attributes or roles rather than based on the identity, which would allow for fine-grained access control and necessary level of confidentiality. At the moment a number of projects provide attribute-based authorization framework, including Shibboleth [13], PERMIS [5], Akenti [15], VOMS [1]. However, these competing solutions are often based on incompatible standards and require users to have different authorization tokens for interoperability with Grids. Although there are continuous attempts to develop a common authorization framework for Globus Toolkit, significant effort is still required to provide an integrated solution.

In this paper we describe our work to integrate the Globus Toolkit and the widely accepted Shibboleth and PERMIS projects in order to provide policy driven role-based access control to Grids. The goal of our project is to integrate the identity federation and attribute assignment function of Shibboleth with the policy based enforcement function offered by the PERMIS access control infrastructure to authorize Grid jobs running with Globus Toolkit v4. Shibboleth is inter-realm attribute-based access control solution already at work in a large number of research and educational institutional thus using it for authorizing Grid jobs will allow leveraging existing security infrastructure whereas policy driven RBAC features of PERMIS will provide the necessary level of fine grained access control and flexibility in assigning and changing users rights.

This paper starts with a brief description of Globus Toolkit, Shibboleth software and an overview of the GridShib research project, as well as the limitations of the current GridShib solution. We then describe PERMIS framework and summarize our work on integrating GridShib and PERMIS projects and the
lessons learnt. We conclude with our plans and the future work.

2. Globus Toolkit

The open-source middleware Globus Toolkit provides all the necessary Grid functionality and supports the development of service-oriented distributed applications and infrastructures. Globus Toolkit is designed to enable applications that federate Grid resources, such as computers, data, services and networks, and is already widely adopted as a means of building large-scale academic and commercial distributed computer systems. The recent version four of Globus Toolkit has seen the convergence of Grid and Web services through the adoption of Web Services Resource Framework standards.

In order to address the security issues arising from the sharing and coordinated use of resources in distributed VOs Globus Toolkit provides the Grid Security Infrastructure (GSI) [18], which is based on the public key cryptography and includes transport-level and message-level security mechanisms [21]. GSI authentication and authorization capabilities are built upon the use of X.509 end entity certificates [19] and X.509 proxy certificates [16], which assert the users identity expressed as the unique Distinguished Name (DN). Proxy certificates allow the user with valid X.509 certificate to temporarily delegate his privileges to another entity thus easing the authorization process providing single sign-on support. X.509 certificates are used for mutual authentication of relying parties if both parties trust the Certificate Authorities (CAs) that signed each other's certificates.

The GSI server-side authorization framework implements the decision engine, which evaluates a chain of authorization schemes – Policy Decision Points (PDPs) in order to determine access rights of the user making a request for a particular Grid service or resource. This chain may also include Policy Information Points (PIPs), which do not return any decision at all and can be used to collect information necessary for the decision making process. Both PDPs and PIPs are classified by the Globus Toolkit as interceptors. However, authorization in GSI is by default based on access control lists (ACLs) located in grid-mapfile, which specify the DNs of the users allowed to access each Grid resource. This approach provides very limited functionality and is inconvenient for the large-scale distributed systems as it lacks necessary flexibility in describing users’ rights. Starting with the version 4 Globus Toolkit GSI uses the SAML standard [14] from OASIS, which also allows for the use of external third party authorization decision services through the SAML AuthorizationDecision protocol callout [20] or the SAML Attribute Request protocol, thus including custom PIPs and PDPs in the authorization chain. This method was used in our project in order to carry out the integration and to enhance the authorization options of Globus Toolkit through the use of Shibboleth and PERMIS.

3. PERMIS

PERMIS (PrivilEge and Role Management Infrastructure Standards Validation) is a policy-based authorisation system being developed at the University of Kent, which uses X.509 attribute certificates stored in an LDAP directory to hold roles/attributes [3]. Given a user’s DN, a resource and an action, it says whether the user is granted or denied access based on the RBAC policy for the resource and the user’s validated roles and attributes.

PERMIS is based on the Privilege Management Infrastructure (PMI) mechanisms (ISO), which use Public Key Infrastructure principles of operation. It introduces the attribute certificates (AC), which maintain a binding between the user’s unique identifier and one or more of his privilege attributes much like public key certificate relate user’s name and his public key. ACs are signed by the Attribute Authority (AA), whilst the root of trust for the PMI is called the Source of Authority (SOA). The discretionary access control is implemented according to the X.509 PMI infrastructure by storing users’ rights in their ACs and by allowing users to issue ACs for the resources they control.

PERMIS uses the role-based access control scheme (RBAC), which means that each user is assigned to one or more roles, typically representing organisational roles, and each role is given a set of permissions in the authorization policy. In order to gain access to a protected target resource a user has to present his roles and the target checks them across the policy in order to make an authorization decision. PERMIS implements the policy-based hierarchical RBAC model, which means that the roles are organised hierarchically with superior roles inheriting the privileges of the inferior ones and a policy is defined which regulates the process of assigning roles to users and permissions to roles, as well as controls the users’ requests to the targets within the policy domain.

The PERMIS PMI architecture consists of a privilege allocation subsystem, which issues the X.509
attribute certificates and is responsible for storing these in the LDAP directory, and a privilege verification subsystem, which authenticates and authorises the users. Authentication is specific to the given application, but authorisation is always performed according to the PERMIS RBAC authorisation policy.

PERMIS provides an API in Java language which controls access to the target resources according to the XML policy [2]. The API caller provides the authenticated name of the user, who is identified by either his LDAP DN or his public key certificate, which is used to retrieve user’s ACs stored in the configured LDAP directory. After that ACs are checked against the authorisation policy and the valid roles are extracted, which are then used to check the user’s requested action. Finally, API returns the “permit” or “deny” authorisation decision to the caller. This approach allows the system to control access to all resources in the domain.

The use of PERMIS for authorising Grid jobs would provide a flexible and efficient way of describing users’ rights and permissions by means of the detailed authorisation policy. However, to provide a fully integrated solution we also need a privacy-preserving method of ascertaining the user’s identity and roles, such as institutional affiliation. Such functionality is provided by the widely adopted attribute-based Shibboleth authorisation software, described below, which was used as an integrated part of the GridShibPERMIS project.

4. Shibboleth

Shibboleth is an Internet2/MACE project providing cross-domain single sign-on and attribute-based authorisation for systems that require inter-institutional sharing of web resources preserving end user privacy. Shibboleth software is already at use today and has been widely experimented with as an inter-realm fine-grained access control solution for the research and education applications. The main idea behind Shibboleth is that instead of login and be authorized at each restricted site users authenticate at their home campuses, which then passes user’s attributes back in a privacy preserving way to the resource providers.

The Shibboleth security protocol is based on SAML (Security Assertion Markup Language) assertions developed by OASIS. The Shibboleth specification [6] is a direct extension of the SAML 1.1 browser profiles [12]. It includes two main software components: the Identity Provider (IdP) and the Service Provider (SP), which are deployed separately. Shibboleth IdP creates and manages user identity and includes four primary components: the Attribute Authority (AA), which releases attributes to the SP Attribute Requester; the Handle Service (HS), which generates random temporary handles to be used in communication with SP thus preserving user’s confidentiality; the directory service; the local sign-on system (SSO). Shibboleth SP controls access to the resources by requesting and consuming necessary attributes from the IdP.

When the Shibboleth SP receives a request from the user for a Shibboleth-protected resource it redirects him to a “Where are you from?” service, which asks the user for the organisation they want to authenticate in. After that the user is redirected to their home authentication system’s Shibboleth IdP Handle Service, which generates a random handle that is returned to the SP. The SP can then use this handle to request user’s attributes from the IdP and according to these attributes, it can decide whether to grant or deny access to the resource. The IdP can use the existing institutional identity management infrastructure, such as LDAP directories, for storing attributes. Shibboleth authentication and authorization session consists of a cookie associated with a security context being passed between the user’s browser and the web server.

Shibboleth provides the efficient means of identity federation and inter-institutional authorization with an emphasis on user privacy protection and management. It can be used to add attribute-based authorization to Globus Toolkit. The GridShib project described later in this paper is currently being developed to allow for the use of the Shibboleth service to authorize Grid jobs in VO context. Our project is partly based on the results of the GridShib project and is carried out in collaboration with the GridShib team.

5. GridShib

GridShib is a research project, currently being undertaken at the University of Illinois, University of Chicago and Argonne National Laboratory that allows for interoperability between the Globus Toolkit and Shibboleth [2]. The first release of the GridShib software consisting of separate plug-ins for Globus Toolkit and Shibboleth has just been made in September 2005. It provides the Globus Toolkit Grid Service Provider with the means to securely request user’s attributes from the Shibboleth IdP. They are collected by a Policy Information Point (PIP) and passed to a Policy Decision Point (PDP) which makes an authorization decision based on the attributes obtained from the Shibboleth attribute authority. This
separation between PIP and PDP allows for the flexible use of the plug-in and was used in our project in order to distinguish between Shibboleth attribute collection carried out by GridShib and decision making implemented by PERMIS.

The large challenge of integrating Globus Toolkit and Shibboleth is that there are currently no unified standards for the attributes transfer from the attribute authority to the relying services and no policies regarding these attributes, thus different IdPs use different name forms and name values to identify the same user – distinguished names in GT4 and opaque handles in Shibboleth. GridShib team had to overcome this incompatibility issue by introducing the DN mapping plug-in for the Shibboleth, which significantly constrained the scalability of the project and is still a major issue that needs to be resolved.

The focus of the GridShib project is to transport SAML assertions from the Shibboleth IdP to the Globus Toolkit-based decision making services [22], thus it provides limited authorization functionality. Consequently it provides limited PDP functionality itself. The GridShib PDP makes the authorization decision by parsing the attribute assertions and comparing them to an access control list that contains the allowed attributes, which is similar to the current grid-mapfile functionality. This leads to users being granted access if they have any single attribute in the access control list. Consequently the PDP is also not capable of making decisions based on dynamically changing conditions, such as the time of day, the amount of resources being consumed, or even on parameters of the user’s request, such as the operation, the requested target or the job priority etc. Furthermore, it is not able to check if the IdP issued the correct set of attributes that it is trusted to issue. The PERMIS decision-making engine on the other hand, is capable of making such decisions, since its PDP is policy driven and can test for arbitrary conditions being fulfilled before access is granted. We will now describe our project, based on the first release of the GridShib project.

6. GridShibPERMIS

In order to carry out the GridShib and PERMIS integration the GridShibPERMIS Context Handler was written as a PDP to be included into the Globus Toolkit authorization framework. As already mentioned above, the Globus Toolkit GSI authorization framework allows including custom PIPs and PDPs implementing authorization schemes into the authorization chain through the callout to external authorization services. This chain with relevant configuration details can be configured at resource, service or container level. The authorization engine is invoked immediately after the authentication of the request is finished and evaluates each PDP and PIP in the order they are specified in the chain. The decision is made on the basis of a deny-override mechanism [10]. Globus Toolkit GSI demands any PDP to implement the interface org.globus.wsrf.security.authorization.PDP and to return permit or deny decision on the basis of subject DN, message context and the requested operation. PIPs in turn have to implement the org.globus.wsrf.security.authorization.PIP interface and must return a set of attribute assertions.

There are two basic methods of assertions transfer which are used for passing information between PIPs and PDPs and for making the callouts – “pull” and “push”. In “pull” mode after the authentication is done the Grid Service Provider requests the attributes from the external domain, that is Shibboleth IdP in user’s own administrative domain for GridShib. In “push” mode attributes are obtained prior to making a request and are pushed to target service at the time of making the request. Currently GridShib provides only the “pull” mode of operation, which is analogous to the normal Shibboleth operation.

The PERMIS API provides three main function calls used in the GridShibPERMIS Context Handler – getPBAAPI, getCreds and decision. The getPBAAPI function initialises the PERMIS PDP and tells the access control decision function (ADF) to read the authorisation policy AC, passing the name of the trusted SOA, address of the LDAP directory and other configuration parameters. The getCreds method is used to pass the user’s LDAP DN to the ADF, which uses it to retrieve all user’s role ACs, which are at this step also validated against policy to check, for instance, that DN is in the valid subject domain. The decision function checks if the requested action on the target resource is allowed for the set of valid roles that the user has, taking into account constraints specified in the target access policy, and returns the authorisation decision.
As the goal of the project was to integrate the alternative PERMIS PDP with GT4, so that it could use the attributes collected from Shibboleth, the Globus authorization chain was configured to invoke the Context Handler after the GridShib PIP. The Context Handler extracts attributes returned as SAML attribute assertions from the Shibboleth IdP, and converts them into the format recognized by PERMIS. This was done by extracting and parsing the name-value pairs, adding the scope information if necessary. It then passes them along with information about the user’s requested action and resource to the PERMIS PDP via its existing Java API. The GridShibPERMIS Context Handler also allows X.509 ACs to be returned, and these are passed to the PERMIS PDP unchanged. Therefore, the PERMIS PDP operates in “push” mode, with the set of authorization tokens provided by the GridShibPERMIS Context Handler. The PERMIS PDP first checks if the IdP is trusted to issue the attribute assertions it has returned, and then it makes an authorization decision according to the configured PERMIS policy and the user’s validated attributes. Figure 1 summarizes the integration scheme.

7. Conclusions and Future Work

The prototype GridShibPERMIS software has been developed and will soon be released for download. It requires Globus Toolkit v4 with both GridShib and PERMIS installed and provides the GridShibPERMIS Context Handler for access control decision making. This has to be configured to be called after the GridShib PIP.

The GridShibPERMIS project also resulted in a research paper, written in co-authorship with Prof David Chadwick and Dr Sassa Otenko, which was accepted for the TERENA Networking Conference 2006 with a 50% acceptance rate. Besides, the project won funding from JISC UK.

The GridShibPERMIS project provides an integrated role-based access control solution for Globus Toolkit. It provides the advantages of both Shibboleth cross-organisation identity federation and attribute management with PERMIS policy driven role-based access controls, giving the necessary functionality, flexibility and fine grained access control for authorizing Grid jobs. However, there are still a number of issues to be resolved in the forthcoming and future releases of the software.

Firstly, there is a problem when using multiple cryptographic service providers with Globus Toolkit, which at first looked to be similar to one of the problems experienced during Globus Toolkit version 3 and PERMIS integration project. However, by carrying out extensive logically coherent testing and research into the problem we figured out that it is not an issue of the GridShibPERMIS integration, but rather appears to be a conflict between the Claymore...
SSL Toolkit used by Globus Toolkit and GridShib and IAIK JCE cryptography library used by PERMIS which results in the private key classes clash when establishing the SSL connection. The cryptography structure of GridShibPERMIS integration explaining the situation is shown in the Figure 2. In PERMIS, X.509 ACs are passed to the PERMIS decision-making mechanism, whilst GridShib uses SAML assertions, which are a string encoding of an XML data type. As a consequence, the digital signature is mandatory in ACs and optional in SAML assertions. Shibboleth and thus GridShib omit the signature, and protect the SAML assertions by an SSL connection between the IdP and the SP. However, it appears that when SSL connection is established with the use of Claymore SSL Toolkit provider it doesn’t take into account that other JCEs may be present in the system and thus crashes after the IAIK JCE security provider has been initialised. It assumes the specific class of the private key; however, as the IAIK JCE already took over, this results in the private key classes clash. The problem didn’t seem to appear in the original Globus Toolkit and PERMIS integration, as IAIK was used after the SSL connection to GT4 was established, however in the GridShibPERMIS integration SSL connection to the Shibboleth IdP is established after the IAIK is loaded, thus resulting in the cryptography error. In fact, executing the client twice, the SSL connection between client and GT4 container fails as well. Thus this is not a problem of GridShibPERMIS integration, but rather an issue resulting from the JCEs clash, which was then discovered in the original Globus Toolkit and PERMIS integration as well by trying to execute the decision-making engine twice. For the results of the tests please consult the testing documentation provided. This problem was resolved by making sure that Signature Verification doesn’t depend on the IAIK JCE, as it is only needed when signing ACs. However, we are still working with the Globus development team to provide a permanent solution to the problem so that multiple cryptography providers could be simultaneously used.

![Figure 2: GridShibPERMIS Cryptography Scheme](image)

The current methods of attribute assertions communication between Shibboleth, Globus Toolkit and PERMIS do not yet support one of the main Shibboleth advantages – pseudonymous access, as user’s DN and roles are currently being released to Grid application in the process of authorization. One of the methods proposed by the GridShib team in order to solve the problem is to include in the authorization
framework the GridLogon service, which would act much like the Shibboleth Handle Service providing the authenticated user with a set of credentials with a pseudonym identifier.

As mentioned earlier, there is currently a scalability problem because the Shibboleth IdP plugin requires each Shibboleth handle to be mapped into the equivalent Globus DN. The Globus team are working on a solution which will use an online CA and newly minted certificates.

There is also a limitation which prevents users with attributes assigned by different AAs from merging them and using them to gain access to a resource. We have a proposal for resolving this which is currently under review.

Another issue is validation of the attribute authorities of a user in the Grid context. As described above, this can be done if the scope domain is included in the attribute values by the IdP, but if it is not, this requires the GridShib PIP to either add it to the attribute values or pass it to the Context Handler so that the PERMIS CVS can validate it. The current limitation is that only one issuer scope domain is supported through the PIP configuration file. It is proposed to make this more dynamic and infinitely extensible by adding the IdP providerId to the authorization request and attribute issuer property to the SAML assertion response. This will be carried out in collaboration with the GridShib team.

On the whole, the GridShibPERMIS project implements an efficient policy driven role-based access control for the use in the attribute-based authorization in Grids. We believe that the results of this project will provide a robust and simple mechanism, having low deployment costs, for enforcing Grid security because it leverages the existing Shibboleth and PERMIS infrastructures to provide policy driven role-based privilege management.

Acknowledgements

Thanks to Prof David Chadwick, the project supervisor, for the project proposal, guidance and support. We are also grateful to Dr Sassa Otenko for the technical advice during the meetings and support with setting up and configuring server-side Shibboleth testbed and PERMIS LDAP directory, as well as help with server-side debugging.

The Authors would like to thank the UK JISC for funding the GridShibPERMIS project. We also thank Tim Freeman and the GridShib development team for their continued cooperation and support.

“Globus Toolkit” is a registered trademark of the University of Chicago.

“Shibboleth” is a registered trademark of Internet2.

References


