Journal of Computer Science 7 (8): 1295-1301, 2011
ISSN 1549-3636

A Stringent Authorization using Principles and Policy for Grid Computing

[1]A.R. Jayasudha and [2]T. Purusothaman
[1]Department of Computer Applications,
Hindusthan College of Engineering and Technology,
Coimbatore, 641032 Tamil Nadu, India
[2]Department of Computer Science, Government College of Technology,
Coimbatore, 641012 Tamil Nadu, India

Abstract: Problem statement: The current information security mechanisms are insufficient to address authorization issues. The access control models today are mostly static and they are not well-suited for the service-oriented environments where information access is dynamic in nature. Traditional authorization security techniques do not directly address these concerns as they primarily use access control lists for authorization, where the user whose name appears in the list is authorized to access the grid with some privileges associated with the names, which requires the resource provider to maintain authorization decisions for every user, which is very time consuming and non-scalable solution. **Approach:** Organizations pass user roles instead of name and date of birth but it used Public Key Infrastructure user certificate for authorization which is inflexible when it comes to open distributed systems (Grid) as it assumes a pre-agreed trust between Service Provider and the Service consumer. Usage of Java authentication and authorization services is performed in a pluggable fashion. It permits the application to remain independent from underlying authentication technology. **Results:** Our implementation provides service providers with full control over authentication and authorization of accounts that access services. Implementation of the proposed technique has proved to be less time consuming and more secured for authentication and authorization as compared to the traditional way of authenticating the users. The Policy Decision Service is envisioned to be used by many Web services protected by their PEPs. **Conclusion:** The model brings out many advantages over traditional identity. It is more flexible and more powerful and is suited for dynamic environments for Web services.

Key words: Public key, authentication technology, underlying authentication, policy file, web services, preferred customers, authorization decision, login module, decision point, traditional authorization, authorization service, login module

INTRODUCTION

The emergence of web service technologies has enabled information systems to interoperate in a platform-independent fashion, promoting unprecedented collaboration and information sharing, often across enterprise and network boundaries. As more organizations and users are becoming interested in using grid computing systems in a variety of application domains, security becomes a key issue (Simmons *et al*., 1991; Stell, 2004; Welch *et al*., 2003; Yagoubi and Slimani, 2007). Public Key Infrastructure assumes a pre-agreed trust between the service provider and the service consumer. However, in an open distributed environment such as Grid systems, resource providers and consumers can join the grid dynamically and these pre-agreements cannot be presumed. The use of JAAS authentication is performed in a pluggable fashion and it makes the applications remain independent of the underlying authentication technology. The Login Modules remain independent of the different types of user interaction. SAML technology provides a way to represent authentication, attribute and authorization decision information in XML. XACML provides XML schema for expressing policies and rules. SAML and XACML are combined to support distributed authentication and authorization. Authorization based on users' credentials is difficult to

Corresponding Author: A.R. Jayasudha, Department of Computer Applications, Hindusthan College of Engineering and Technology, Coimbatore-641032 Tamil Nadu India

manage and PKI is inflexible when it comes to Grid. Our work uses XACML, a general purpose access control policy language. It provides syntax in XML to define action (request) rules for subjects (users) and targets (resources). It describes both access control policy language and a request/response language. Access control policy language is used to express access control policies (who can do what, where and when). The request/response language expresses queries about whether a particular access should be allowed.

MATERIALS AND METHODS

Authorization approaches: Currently existing Grid authorization consists of access control lists (grid mapfile) which is not scalable as it requires the Resource Provider to maintain authorization state for every user which is time consuming. User Credential for one gird cannot be used directly to access resources at another (Shamir, 1979). A user needs to maintain multiple credentials if he/she wants to use multiple grids.Recent grid security trends try to overcome this problem by introducing a new authorization technique, which supports policy based authorization, which means authorization decisions are not based on the users' identity but on the policies generated and sent to the service providers. PKI presumes a pre-agreed trust between the service provider and the service consumer. In an open distributed environment such as grid systems, service providers and resource consumers can join the grid dynamically. PKI certificate contains a set of user attributes for determining authorization. These attributes have to remain static. If the user changes his/her role, some attributes could change as well (Ito *et al*., 1987). Thereby the certificates need to be invalidated and revoked. Hence, a user has to have multiple certificates for each service and remember which certificate to use, which is infeasible (Benaloh and Leichter, 1989). Implementations of authorization in recent years have largely adopted the XML based SAML and XACML standards for authentication and authorization. Combination of SAML and XACML proves more beneficial.

Drawbacks of role based authorization: Individual actors called Entities are defined by public keys. Let A,B,C,D,E range over entities. Each entity can create arbitrary number of Roles in a namespace local to the entity denoted A.r of roles. Suppose a Hotel H offers a room discount to certain preferred customers, who are members of H.preferred. The policy of H is to grant a discount to all of its preferred customers in H.preferred as well as to members of certain organizations. H defines a role H. orgs that contains the public keys of these organizations. Into that role H places, for example the key of SSS, an association. The credentials are summarized as:

H.discount← H.preferred
H.discount←H.orgs.memb
H.discount←SSSAt a later time, a special plan is created to encourage travelers to
stay at H. A decision is made that all members of SSS are automatically preferred customers.
H.preferred← SSS.memb
If X is a member of SSS. She has a credential
SSS.memb ← M

X can prove in two different ways that he is authorized for discount in two distinct ways. One he is a member of H.orgs and the other a preferred customer of H. Practical considerations may motivate H's decision about which proof to use.

Implementation of JAAS: JAAS uses an updated technology that is plugged without requiring modifications to the application. It uses different underlying technologies such as Kerberos. There are different ways of communicating with the user. The Login Module performs authentication by remaining independent of the different types of user interaction. Figure 1 demonstrates the architecture of JAAS. For communication KDC generates a session key which is used for secure interaction. The Login Context constructs the configured login module and initializes it with new subject and callback handlers. In order to authenticate a user, javax.security.auth.login.Login Context is required.

Login contextlc:

NewLoginContext(<config file entry name>, <CallbackHandler to be used for user interaction>)

The LoginModule invokes a javax.security.auth.callback.Callback Handler to perform the user interaction and obtain the requested information, such as the user name and password. Snippet1 shows different ways of interacting with the user. The calling application can subsequently retrieve the authenticated Subject by calling the Login Context's getSubject method. Figure 1 shows the architecture of JAAS.

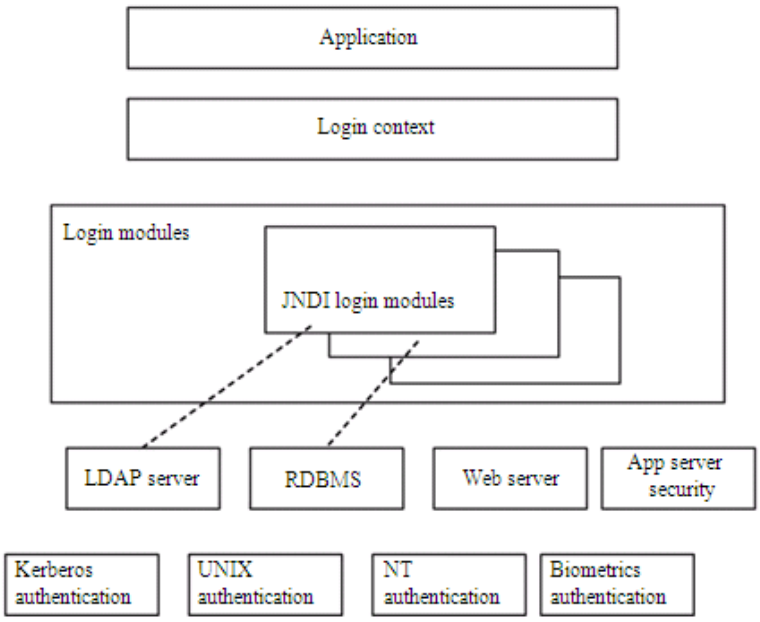

Fig. 1: JAAS Architecture

```
if (callbacks[i] instanceofTextOutputCallback) {

 // displays the message according to the specified type
TextOutputCallbacktoc
(TextOutputCallback)callbacks[i];
switch (toc.getMessageType()) {
caseTextOutputCallback.INFORMATION:
System.out.println(toc.getMessage());
break;
caseTextOutputCallback.ERROR:
System.out.println("ERROR: " + toc.getMessage());
break;
caseTextOutputCallback.WARNING:
System.out.println("WARNING: " + toc.getMessage());
break;
default:
throw new IOException("Unsupported message type: "
+ toc.getMessageType());
 }
```

Snippet1 for authentication.

The Login Configuration specifies that the module which uses Kerberos for authentication is required to have success for authentication:

Let K_a bethe master key for A shared by A and KDC. K_{ab} session key shared by A and B. T_b Ticket to use B. K{data}→ data encrypted with key K
A - $K_a\{K_{ab},B\} \rightarrow K_{ab}$ $K_b\{K_{ab},A\}$-B

Performs mutual authentication to prove that A and B know each other. User logs in with the granted identity, based on which KDC generates password for further communication for accessing services by granting a ticket.

Implementation of JAAS Authorization: JAAS authorization extends the java security architecture that uses security policy to specify the access rights to execute. The permissions are granted based on code characteristics. A subject is created when a user is authenticated. The Subject carries the identity that distinguishes it from other Subjects. The purpose of the Subject is to represent the authenticated user. A Subject is comprised of a set of Principals, where each Principal represents an identity for that user. For example, a Subject could have a name Principal ("SS") and a Social Security Number thereby distinguishing this Subject from other Subjects. A Policy file is generated that includes one or more principal fields:

```
grant codebase “file:./trialact.jar”, Principal
trial.principal.trialPrincipal “tsttrial”{
        permissionjava.util.PropertyPermission
“java.home”, “read”;
        permissionjava.util.PropertyPermission
“user.home”, “write”;
       permissionjava.io.FilePermission “”, “read” };
```

Snippet2 for policy file

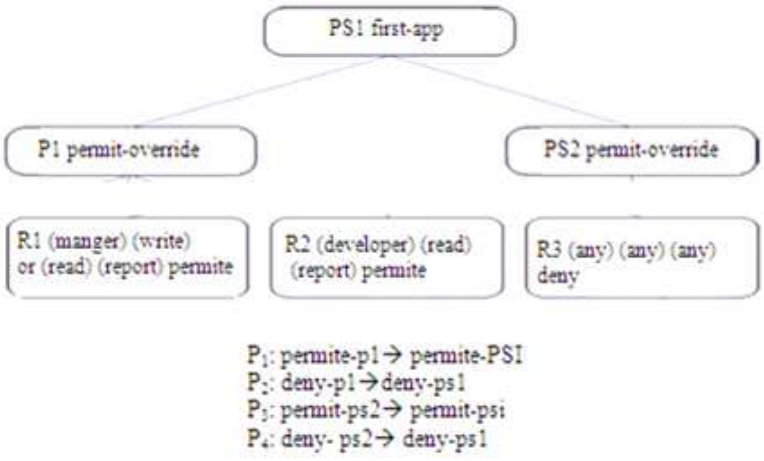

Fig. 2: Permit/deny override

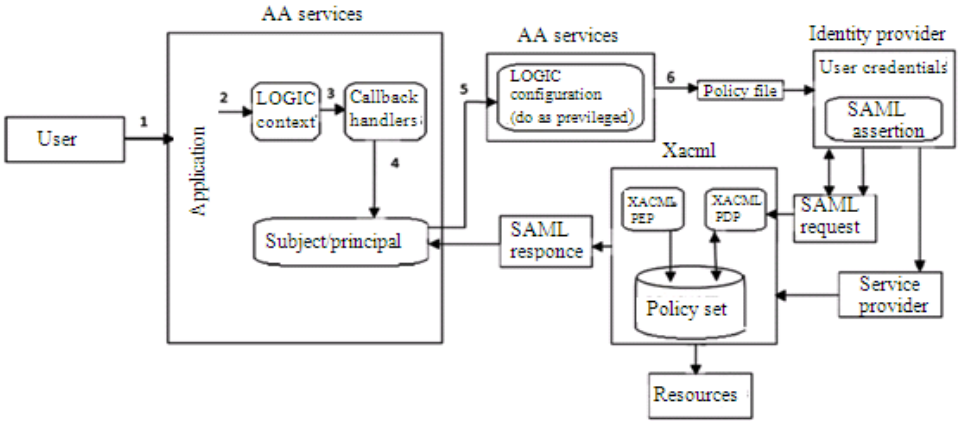

Fig. 3: Architecture of JAAS with SAML and XACMl

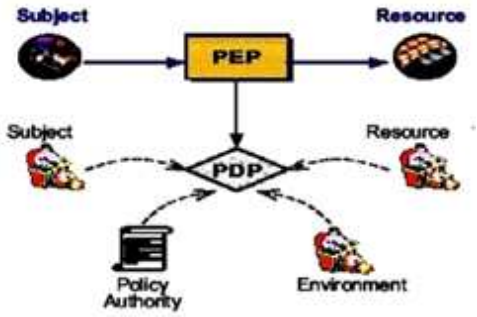

Fig. 4: Architecture of JAAS with SAML and XACM

Permissions can be granted in the policy to specific Principals. After the user has been authenticated, the application can associate the Subject with the current access control context. For each subsequent security-checked operation (a local file access, for example), the Java runtime will automatically determine whether the policy grants the required permission only to a specific Principal and if so, the operation will be allowed only if the Subject associated with the access control context contains the designated Principal (Brickell, 1989). A subject is associated with access control after it is authenticated and authorized. Figure 2 shows the permit and deny override. The doAs method is called with an authenticated subject. It associates the subject with the current access control and invokes run method from the action. The run method implementation contains all the code to be executed as the specified subject:

Privileged Action act = new trialAction();
Subject.doAsPrivileged(subject,action,null);
Snippet 3 for Privileges.

XACML formulation:

Implementation of SAML and XACML: The Security Assertion Markup Language is an XML

framework for exchanging authentication and authorization information (Zhang *et al*., 1999). This security information is expressed in the form of assertions about subjects, where a subject is an entity that has an identity within a security domain. Assertions can convey information about authentication acts performed by subjects, attributes of subjects and authorization decisions about where subjects are allowed to access a certain resources. Our XACML based policy management and authorization system allows a authoritative entity to create, modify and package resource policies. The Policy Decision Point (PDP) answers authorization queries based on the resource policies:

A Rule is represented as
Pre(r) → Con(r)
Rule Set = Strict-rulesχDefeasible-rules

Policy Enforcement Point: The Policy Enforcement Point (PEP) is responsible for requesting authorization decisions and enforcing them. Figure 3 and 4 shows the flow of sequences among the subjects and the resources. In essence, it is the point of presence for access control and must be able to intercept service requests between information consumers and providers. Although the diagram depicts the PEP as a single point, it may be physically distributed throughout the network. The most important security engineering consideration for the implementation of a PEP is that the system must be designed such that the PEP cannot be bypassed in order to invoke a protected resource:

```
<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<Resource>
<ResourceMatchMatchId=
"urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValueDataType=
"http://www.w3.org/2001/XM
#string">MainServer</AttributeValue>
<ResourceAttributeDesignatorDataType=
"http://www.w3.org/2001/XMLSchema#string"
 AttributeId=
"urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
<AnyAction/>
</Actions>
</Target>
```

The Policy Decision Point (PDP) is responsible for evaluating the applicable policies and making the authorization decision (permit or deny). The PDP is in essence a policy execution engine. When a policy references a subject, resource, or an environment attribute that is not present in the request, it contacts the appropriate AA to retrieve the attribute value(s).

RESULTS AND DISCUSSION

User authorization: XACML implementation is used as a part of the authorization system. Their main function is to bind the XACML schemato javarepresentations, handles the attributes and marshals the contents to XACML request format. In the most general form, a Policy Rule that decides on whether a subject s can access a resource r in a particular environment e, is a Boolean function of s, r and e's attributes:

Rule: can_access (s, r, e)
$\int$(ATTR(s), ATTR(r), ATTR(e))

Given all the attribute assignments of s, r and e, if the function's evaluation is true, then the access to the resource is granted; otherwise the access is denied. A Policy rule base or Policy Store may consist of a number of policy rules, covering many subjects and resources within a security domain. The access control decision process in essence amounts to the evaluation of applicable policy rules in the policy store.

Results obtained using JAAS has been compared with the results of the earlier authenticating system. It proves less time consuming and more secured:

```
Enter the user name: JAVA
Enter the pwd: object oriented programming language
Inside get dbconnection
Elapsed time is→ 6 ms
Bash-3.004 java time diffcheck
Enter the user  name system1
Enter the pwd: systemadminone
Inside getdbconnection
Elapsed time is→ 6 ms
Bash-3.004 java time diffcheck
Enter the user name: user name
Enter the pwd:  password
Inside getdbconnection
Elapsed time is→ 4 ms
Bash-3.004 java time diffcheck
Enter the user name: cbe
Enter the pwd:  hindusthan 0 coimbatore
Inside getdbconnection
Elapsed time is→ 4 ms
Bash-3.004 java time diffcheck
```

Table 1: Measuring time for distributed environment

Password length	Time (milliseconds)
21	4ms
26	5ms
36	6ms
16	6ms
8	4ms
23	4ms

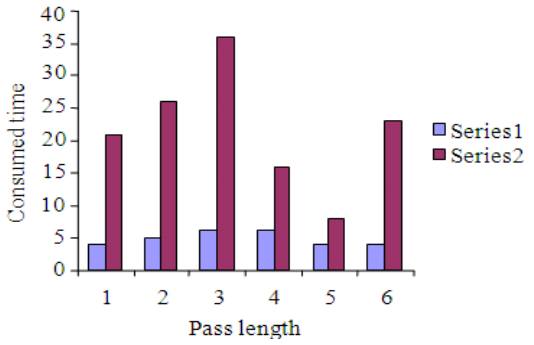

Fig. 5: Time for distributed environment

The conventional way of authentication is analyzed and compared with the newly developed JAAS authentication. Table 1 show the time measured during conventional authentication. Figure 5 demonstrates the varying time consumed during authentication.

Algorithm:

- The main assumption is the Service providers and the Identity providers trust each other within in a VO
- The application sends the authentication and authorization service its request for authentication
- Login context and callbacks determine the technology to be used for authentication
- Successful authentication generates a subject/Principal with SSN number which is its unique identity
- Users are authorized based on the SSN and a policy file is generated
- Policy file carries the identity of the user with the resource/services that he/she is authorized for
- Depending on the users' request, a SAML assertion is send to the Identity Provider
- The Identity Provider after matchmaking forwards to the service Provider
- Based on the assertion, a XACML request is marshaled to a file. It also contains Policy Enforcement Point(PEP) and Policy Decision Point (PDP)
- XACML PDP generates authorization decision statement and forward the decision to the XACML PEP
- Finally, the response is send to the subject in the form of a SAML response

CONCLUSION

This study analyzed the requirement of authorization services and proposed the use of JAAS and combination of SAML and XACML as a basic mechanism for implementing authorization services in large scale distributed computing systems.

While pointing out the desirable features of the complex authorization policies, the study also discussed the practical limitations of traditional authorization in such environment and designed an new authorization service architecture for VO. authorization service for VO. Further research includes satisfactory of protocol authentication and visualization monitor to the execution effect of authorization scheme.

REFERENCES

Benaloh, J. and J. Leichter, 1989. Generalized secret sharing and monotone functions. Adv. Cryptol., 403: 27-35. DOI: 10.1007/0-387-34799-2_3

Brickell, E.F., 1989. Some ideal secret sharing scheme. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in cryptology, Springer-Verlag New York, Inc. New York, NY, USA., 105-113. ISBN: 3-540-53433-4

Ito, M., Saito, A. and T. Nishizeki, 1987. Secret sharing scheme realizing general access structure. Elect. Commun. Jpn., Part III: Fundamental Elect. Sci., 72: 56-64. DOI: 10.1002/ecjc.4430720906

Johnston, W., S. Mudumbai and M. Thompson, 1998. Authorization and attribute certificates for widely distributed access control. Proceedings of the IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Jun. 17-19, IEEE Xplore Press, Stanford, CA., pp: 340-345. DOI: 10.1109/ENABL.1998.725715

Jayasudha, A.R., 2010. Grid Scheduling using Differential Evolution (DE) for solving multi-objective optimization parameters. Int. J. Comput. Sci. Eng., 2: 2322-2327. ISSN: 0975-3397

Keahey, K. and V. Welch, 2002. Fine-grain authorization for resource management in the Grid environment. Proceedings of the 3rd International Workshop on Grid Computing (IWGC'02), Springer-Verlag, London, UK., pp: ISBN: 3-540-00133-6

Shamir, A., 1979. How to share a secret. Commun. ACM, 22: 612-613. DOI: 10.1145/359168.359176

Simmons, G.J., W. Jackson and K. Martin, 1991. The geometry of shared secret schemes. Bull. ICA, 1: 71-88. http://pure.rhul.ac.uk/portal/en/publications/the-geometry-of-shared-secret-schemes%2877a41a12-7060-489e-8ca2-a6e9822932d5%29.html

Stell, A.J., 2004. Grid Security: An Evaluation of Authorization Infrastructures for Grid Computing. MSc Dissertation, University of Glasgow. www.nesc.gla.ac.uk/projects/etf/MScProj.pdf

Welch, V., F. Siebenlist, I. Foster, J. Bresnahan and K. Czajkowski *et al*., 2003. Security for grid services. Proceedings 12th IEEE International Symposium on High Performance Distributed Computing, Jun. 22-24, IEEE Xplore Press, pp: 48-57. DOI: 10.1109/HPDC.2003.1210015

Yagoubi, B. and Y. Slimani, 2007. Task load balancing strategy for grid computing. J. Comput. Sci., 3: 186-194. DOI: 10.3844/jcssp.2007.186.194

Zhang, C.R., K.Y. Lam and S. Jajodia, 1999. Scalable threshold closure. Theoretical Comput. Sci., 226: 185-206. DOI: 10.1016/S0304-3975(99)00072-9