An Access Control Model for a Grid Environment Employing Security-as-a-Service Approach

E.K. Olatunji, M.O. Adigun, and E. Jembere

Abstract

There is a continuous effort at addressing security challenges of large scale service oriented computing (SOC) infrastructures like grids. A lot of research efforts towards development of authentication and authorization models for grid systems have been made because existing grid security solutions do not satisfy some desirable access control requirements of distributed services; such as support for multiple security policies. However, most of these security models are domain and/or application specific. Domain/application-specific approach to providing security solution is a duplication of effort, which also increases the cost of developing and maintaining applications. This paper presents the design of an access control model for grid-based system that employs security as a service (SecaaS) approach. By SecaaS approach, each atomic access control function (such as authentication, authorization) will be provided as a reusable service that can be published and subscribed to by different grid entities. In this approach, each admin domain will no longer need to have its own domain-specific access control logic built into it. Whenever an access control service is required the domain administrator subscribes to this service from SecaaS. This approach has a number of benefits, including making changes to security policies dynamically on the fly.

Keywords

Access Control • Administrative Domain • Grid Computing • Security • Service provider

Introduction

Grid computing enables the use and pooling of computers and data resources to solve problems that are far too big and complex for any single super computer to complete within a reasonable time frame. The goal of grid computing is the provision of flexible, secure, and coordinated resources sharing among dynamic collections of individuals, institution and resources distributed over heterogeneous wide area network.

E.K. Olatunji (✉) • M.O. Adigun • E. Jembere
Department of Computer Science University of Zululand, Kwadlangezwa 3886, South Africa
e-mail: aekolatunji@gmail.com; madigun@pan.uz.ac.za; ejembere@gmail.com

This computing paradigm involves an evolving set of open standards for web services and interfaces that make services or computing resources available on the internet [1]. Application areas and examples of grid projects are detailed in [2] and [3].

Security, however, is a big challenge in a large scale distributed system like grid, as it involves the federation of multiple heterogeneous, geographically dispersed autonomous administrative domains [4]. The peculiar characteristic of a grid environment presents unique security challenges that are not addressed by traditional client-server distributed environments [5, 6]. This security challenge has been attributed to be one of the biggest obstacle to wide scale adoption of grid computing [7], despite its many benefits.

The first and more notable security architecture for grids was the one proposed by Foster et al. as cited in [8].

T. Sobh and K. Elleithy (eds.), *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*, Lecture Notes in Electrical Engineering 313,
DOI 10.1007/978-3-319-06773-5_15,

The architecture addresses several unique grid security requirements. Security requirements within the grid environment are driven by the need to support scalable, dynamic, and distributed Virtual organization (VO), which is a collection of diverse and distributed individuals and organization that seek to share and use diverse resources in a coordinated fashion.

The core security services of SOC systems like grids, including authentication, secure communication, authorization and auditing, should be provided on an end-to-end basis. That is data, services and other resources must be protected over the entire path of requests and responses as they travel through the system [9]. Additional and peculiar securities features that need to be supported by security infrastructure for grid and web services, such as single sign on, multiple security policies, etc., are detailed in [6] and [10].

A number of grid security infrastructures (such as GSI, CRISS, PERMIS, etc.) have been developed because existing security solutions for client-server distributed systems do not address the peculiar security challenges of a grid environment [5, 6]. According to some scholars, these grid security solutions do not adequately satisfy some desirable access control requirements of distributed web services, such as support for multiple security policies, support for fine grained access, etc. [4, 6, 10, 11].

This inadequacy of existing Grid security infrastructures has motivated the proposal, design and/or implementation of a number of other, most often domain/application-specific, authentication (aN) and authorization (aZ) models and infrastructures for grid systems by [4, 6, 10–12]. The problem with this approach of providing domain/application—specific security solution is that of duplication of effort, as well as increase in the cost of developing and maintaining applications. This problem was also noted by [13]. One of the aims of this research effort is to employ an alternative approach to providing access control solutions that is not domain or application specific for a grid environment.

A better method at addressing security issues in grid system, we believe, will be to employ SecaaS approach for satisfying access control requirements of distributed services in a grid environment. By this approach, each atomic security function will be provided as a reusable service to be published and can be subscribed to by different grid entities. In this approach, each admin domain will no longer need to have its own domain-specific access control logic built into it Whenever an access control service is needed the domain administrator subscribes to this service from SecaaS.

Our proposed security model will be constructed to be independent of middleware and service providers/resources owner unlike in the existing models. The model also aims at providing multiple mechanisms for aN and aZ; thus the grid entities will have the priviledge of specifying their preferred mechanism for aZ and aN. Our proposed solution will also be web services security standard compliant.

The SecaaS approach has the following potential benefits:

- Great reduction in the cost of developing and maintaining applications as well as enabling developers to concentrate on the logic of applications
- Developing and maintaining security services at very few points
- Changing of security policies dynamically on the fly at no extra cost
- Flexibility for Service Providers in making use of their preferred access control mechanisms as multiple mechanisms will be supported
- Scalability
- Support for fine—grained access control
- Support for multiple security policies
- Being standard-based

The remaining part of this paper is organized as follows: second section provides a brief review of related works, while components of a typical grid environment are described in the third section. A brief description of existing approach to providing access control in a grid environment is given in the fourth section. The design and description of our proposed SecaaS model are the focus of the fifth section. Finally, the sixth section concludes the paper and gives direction for future work.

Related Work

Review of literatures revealed that a lot of effort has been made in attempt to address security challenges of large scale distributed system like grid. Jie et al. [5] and Singh, Singh & Kaur [4] carried out a review of a number of projects and models that have been carried out in an effort to address authorization and access control related issues in one form or the other. Among these projects are CARDEA, CRISIS, Gridship and Legion [4]. Others include GridshipPERMIS, and Internet2Shiboleth [5].

Security infrastructures/technologies like Globus GSI, Kerberos and Athens [5] are common authentication infrastructures for grid systems. However, each of them makes use of only one mechanism for authentication; none has support for multiple authentication mechanisms. Kerberos is not explicit in provision of single sign-on feature, while Athens has no support for delegation. Support for these features are desirable in grid security infrastructures [4, 5]. In the same vein, authorization infrastructures like CAS [5], VOMS [4], PERMIS and Akenti [5] have provision for only one authorization model, using either user name, a user group, a user role, user attribute, etc. Ideally, service providers only need to determine what type of access to grant to different categories of users and then leave the authorization infrastructure to enforce the policies [5].

A number of researchers, including [4, 10, 11], have also observed some of the inadequacies of many of the existing grid security infrastructures to satisfy some authentication and access requirements of distributed services and have been motivated to propose, design and implement other authentication and/or authorization frameworks and models for distributed services.

For instance, Squicciarini et al. [12] proposed and developed an authentication (aN) framework which supports multiple aN mechanisms combined through aN policies and on the association of aN requirements with the protected resources. They were motivated by the fact that protected resources of a system may require different aN strengths for different users wishing to access them; and that the existing approaches to 'continuous aN' were not expressive enough to support fine-grained aN policies. Their framework does not take care of grid aN requirement such as mutual authentication. Furthermore, it is to be weaved in to the application and thus it is application specific. In addition, aN function is not being offered as a service.

An authorization (aZ) framework that can support multiple policies in the Globus Tookit 4 was designed and constructed by [11]. This was in recognition of the fact that aZ mechanisms in Grid computing platform needs to support multiple security policies and have the flexibility to support dynamic change in security policies. These features are not present in the existing aZ frameworks for grid System.

A policy–based aZ and access control model for a grid environment was designed and implemented by [4]. This was attempted because, according to them, existing traditional client/server based distributed system did not address the peculiar aZ and access control challenge of a large service-oriented computing like grid with many different domains. The model however does not provide for context aware access control mechanism which is also a desirable features of a large scale distributed system. Unlike our model, their aZ model will have to be incorporated in the system of a service provider of a particular domain.

Ekabua & Adigun [10] designed a GUISET-driven aZ framework because, according to them, existing aZ frameworks for grid systems are not suitable or applicable to GUISET—a grid based infrastructure. Their framework was specifically designed for GUISET environment. Our proposed approach of offering security solution in grid environment will not require each application or domain to have its own security subsystem.

The following are note worthy in the works of all these reviewed scholars. They were motivated to carry out their work because existing security solutions for grid are not adequate in providing some desirable access control requirements for web services in a Grid environment. Secondly, the proposed and/or prototyped aZ framework are application or domain specific. Domain-specific approach to providing security solution results in two key related problems: replication of effort as well as increase in application development and maintenance cost. These problems were also noted by [13].

Our intended method of addressing these problems is by employing 'security as a service' (SecaaS) approach in which each key access control security functions will be provided as a service that would be published and can be subscribed to by any grid entity. Service providers will no longer need to bother themselves with implementing the logic of their own detail domain/application specific security solutions for access control. Our proposed security model will be constructed to be independent of middleware and service providers/resources owner unlike in the existing models.

Components of the System

An access control security system can be defined as a system that grants or denies access to protected resources or services (R/S) based on the access rights/privileges of the requesters with respect to the security policies of the owner of R/S. In order to make the system model being presented understood, the following are some of the components or entities that interact together in the system.

a. **Service Requester** (**SR**), (which is more commonly referred to as Subject (SU)), is an entity that wants to make access to a protected R/S in the system. This can be a grid user, a service or any other entity acting on the behalf of a user/service. A SR is denoted by a *rounded rectangle*.
b. **Service** (**S**) is a piece of code that provides some functionality and can be appropriately accessed by SRs. Services are published in the environment and can be discovered and subscribed to by SRs. Service is represented by a *triangle*.
c. **Resource** (**R**) is an object that can be appropriately accessed and used by SRs. Appropriately used in the sense that SR must need to comply with the policies (rules and regulations) guiding the use of both services and resources. Typical resources include CPU, storage, data, applications, scientific instruments, etc. Resources are accessed by SR through services. Thus a resource is also a service in this sense. Resources are denoted by a *square*.
d. **Security Service Policy** (**SeP**) is a set of rules and regulations associated with the use of a service (or a resource). A SR must comply with the policy before access is granted to the service. SeP is represented by an octagon.
e. **Administrative Domain** (**AD**). This is the collection of Service Requesters, Service Providers (SP), Services and

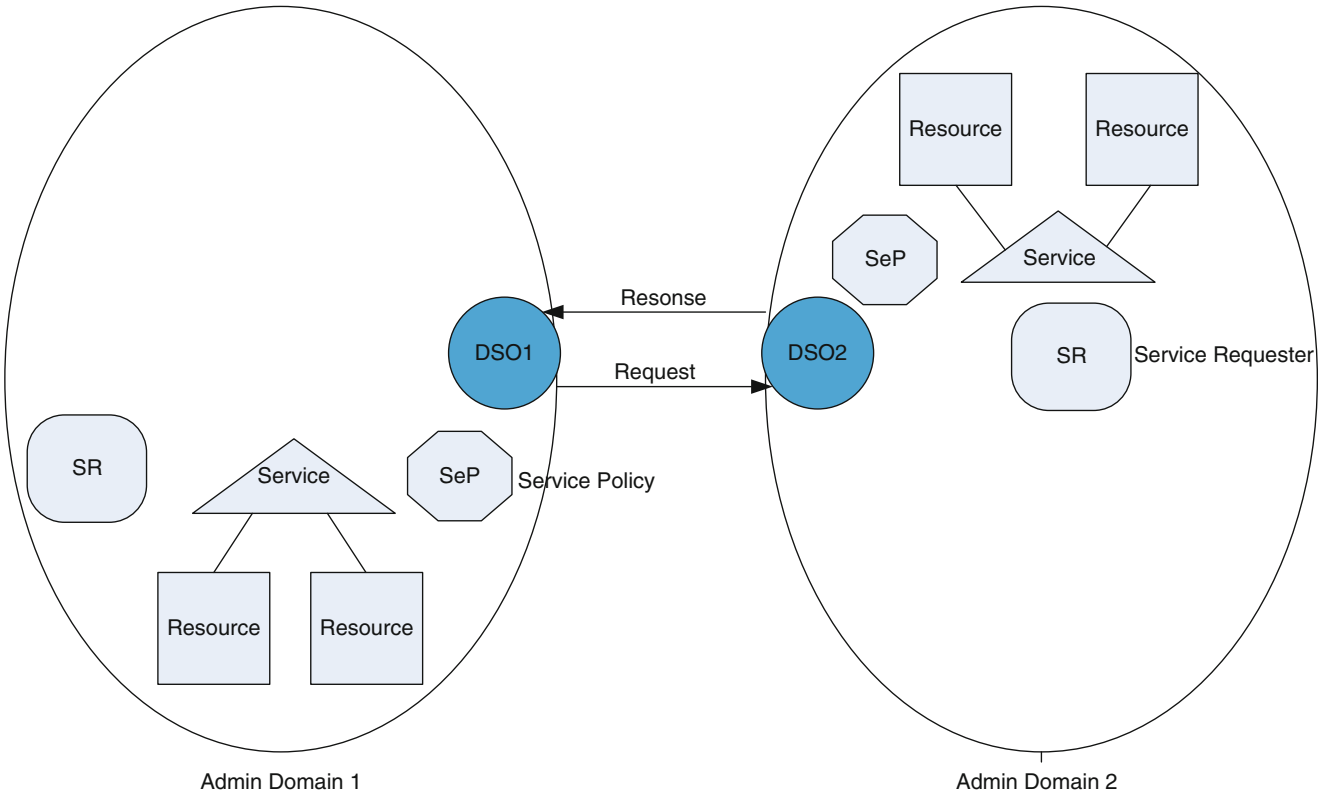

Fig. 1 A grid system consisting of two admin domains together with some of their entities (A SR from AD1 requests for service from AD2)

Resources that are bound by a common but unique domain policy (DP). Domain policy is a set of rules and regulations that an entity must abide with in order to belong to the AD. An AD is depicted by a *big oval* (egg-like shape) or *ellipse*.

f. **Service Providers** (**SP**) are also members of a particular AD that own services and resources which are exposed within a grid environment. They provide service by publishing their services for SRs to discover and subscribe to them. At any time in a grid environment, a SP can transit to a SR as the need arises.

g. **Policy Database** (**PD**). This is a repository for temporary storage of all relevant policies provided by domain security officer (DSO) when requesting for any of the security services from SecaaS. The policies are the one that are applicable to SR and services/resources. In non-SecaaS based security system, all the policies of an AD are stored almost statically in the policy database for the policies.

h. **Domain Security officer** (**DSO**). Each administrative domain has an administrator or a security officer who carries out necessary access control checks for requested R/S by SRs. Access control check is carried out even if an SR is making access to R/S in its home domain. Access request for R/S outside one's home domain is passed through the home DSO to the DSO of the target admin domain. In our proposed model, DSOs subscribe to access control services of SecaaS.

In a typical grid environment, depicted in Fig. 1, these components/entities interact with one another in a complex and dynamic manner. Figure 1 shows a grid system consisting of two admin domains and some of their entities. In the diagram/figure, Service Requester, Services, and Resources are respectively represented by rounded rectangle, triangle and a square while SeP and AD are denoted by an octagon and an oval shape respectively.

As can be seen in Fig. 1, every request for a R/S goes through the DSO who carries out necessary security/access control checks. In the proposed model, DSO carries out access control checks by subscribing to the services of SecaaS.

The Existing Approach of Providing Security in Grid Systems

In the existing approach of providing security solution, each AD has an entity equivalent of DSO, who carries out necessary access control checks for all requested R/S by SRs. Request to access R/S within one's domain is passed through

the home domain's DSO. For instance, both AD1 and AD2 in Fig. 1 have their own domain-specific security infrastructure. Any SR from AD1 attempting to access R/S from its home domain would have to be checked for access rights to his desired R/S. However, if a SR from AD1, for example, wants to access R/S in AD2, the request is passed through RS′ home DSO, the access rights of RS′ home domain (AD1) will be vetted by AD2. Of course, the access rights of a SR from AD1 is necessarily a subset of the access rights granted by AD2 to SR/Subject of AD1. Access to the requested R/S is granted if and only if SR from AD1 complies with the security policies associated with the requested R/S in AD2; otherwise, access is denied.

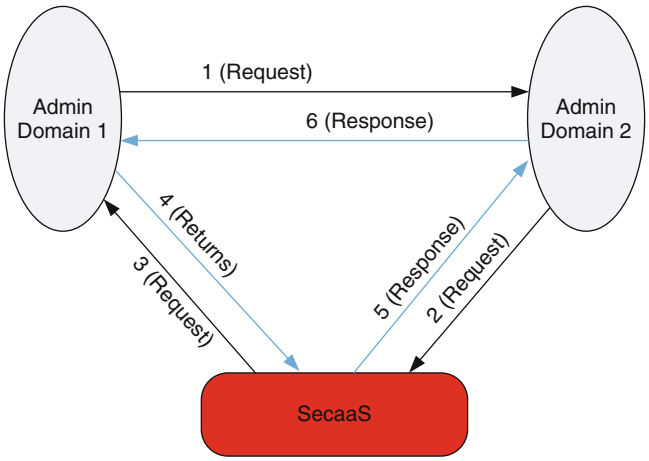

Fig. 2 Request-Response between two Admin Domains (with SecaaS). *Notes* 1. Request for Resource/Service from Admin Domain1. 2. Request for a service from SecaaS by Admin Domain2. 3. Request for security tokens of the requester of Resource/Service. 4. Security tokens of Requester returned to SecaaS. 5. Response from SecaaS Admin Domain2. 6. Response from Admin Domain2 to Admin Domain1

Description of SecaaS Approach

Describing the SecaaS Model

In the proposed SecaaS approach, each AD does not have its own domain-specific security (access control) logic built into it, unlike in the existing approaches. Although each AD still has an entity that serves as DSO. For any domain to carry out access control checks, the domain subscribes for access control services from SeccaS, which performs appropriate access control checks (e.g., aN or aZ checks) for the domain, and returns a 'Grant' or 'Deny' access response.

For SecaaS to carry out requested security/access control checks, the security policies associated with the requested R/S are part of the parameters to be supplied by the domain requesting for security/access control services. These policies are evaluated and a response of 'Grant' or 'Deny' access is returned to the domain requesting for security service. If policy evaluation results in a 'grant access', access will be granted to the SR of the R/S, otherwise access is denied.

All request for R/S goes through the security officer of each domain (DSO). If the SR and requested R/S belong to the same domain, their DSO requests for security service from SecaaS, even if SecaaS provider also belongs to the same domain. However, if the requester and the requested R/S belong to different ADs, the DSO of SR will send request for R/S to the domain of the requested R/S. The DSO of the requested R/S will intersect this request and then request for appropriate security/access control services from SecaaS. Figure 2 is a schematic diagram of the proposed SecaaS approach. It shows the flow of request and response for R/S between two ADs.

In this Fig. 2, when a SR from AD1 through its DSO, for example, requests for R/S from AD2, the DSO of AD2 intersects this request, and carries out access control checks on the SR by subscribing to the services of SecaaS. When SecaaS receives a request for acess control checks on a SR from AD2, it will request for the security tokens (e.g. authentication tokens) of the SR from the SR's AD (i.e., AD1). SecaaS will then carry out necessary access control checks by evaluating the request against the security policies associated with the requested R/S and then return a 'Grant' or 'Deny' access response to AD2. Based on the security decision received from SecaaS, AD2 will either grant or deny access to the requested R/S. Section 'Description of Components of SecaaS Model' is a brief description of relevant components of our SecaaS architecture.

In addition to eliminating the need for every administrative domain to have its own in-built access control logic, the SecaaS approach of providing access control to protected R/S in a grid environment facilitates dynamically making changes to access control policies on the fly.

Description of Components of SecaaS Model

The SecaaS model as was initially proposed by Bertino et al. [13] is as shown in Fig. 3. However, the model has not yet been investigated nor implemented.

In the reference architecture shown in Fig. 3, it is assumed that the process of providing access control to a protected resource and/or services (R/S) encompasses and begins with identification, followed by authentication and then by authorization.

According to the principles of SOA, each component of SecaaS is a service. Furthermore, by following the XACML model of OASIS [14] each component in SecaaS is made up of two distinct elements, the PDP and the PEP; which are

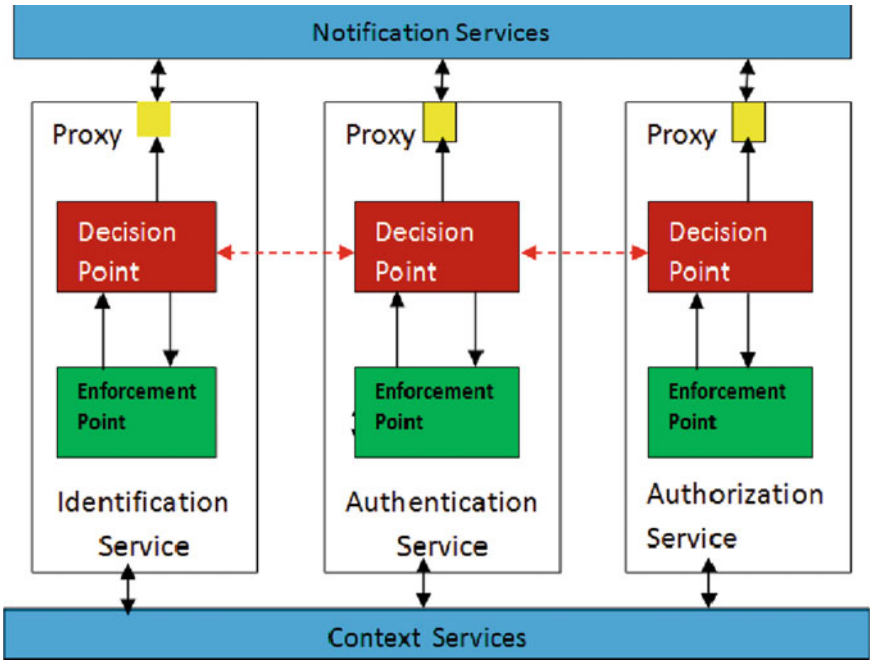

Fig. 3 SecaaS reference model [13]

driven by the relevant security service policies. According to the XACML model, separating the decision point from enforcement point makes it easy to update decision criteria on the fly when the governing policies changes.

The policy decision point (PDP) in the SecaaS reference architecture (see Fig. 3) is the point at which security decisions are made based on the security policies associated with protected resources/services (R/S). Such a decision point for a given security service (e.g. authentication service) may need to interact with another security service's decision point in order to reach a security decision. The policy enforcement point (PEP) is the point where access is granted or denied based on access decisions received from PDP.

The notification Service is included in SecaaS framework so that it can help in maintaining a coherent state of security information. Particularly, it is meant to notify any of the security services in SecaaS of any event relevant to it. For example, in healthcare services, when a Medical Doctor resigns, an identity event can be generated. The context service component is to help keep track of and make available relevant shared contextual information among the security services.

Procedure of Access Request for Resources/ Services in a Typical Grid Environment Using SecaaS Approach

In this section, a scenario in which a requester in one AD (e.g. AD1) is requesting access request to R/S in another AD (e.g. AD2) is described. Figure 4 shows this scenario. The procedure for requesting for R/S in this scenario is as follows:

1. A SR from AD1 makes access request through its home DSO (DSO1) for a R/S in AD2.
2. DSO1 Passes the request to AD2. The request is intersected by DSO2.
3. DSO2 subscribes for access control service from SecaaS in order to ascertain the access rights/priviledges of the SR
4. SecaaS requests AD1 to send the security (e.g. authentication) tokens of SR
5. Security tokens of SR is sent by AD1 to SecaaS.
6. SecaaS engine retrieves security policies associated with the R/S, and evaluate these against the request.
7. Security decisions (grant or deny access) made by SecaaS is conveyed to the security officer of AD2 (DSO2).
8. DSO2 grants access request to the R/S concerned because security decision provided by SecaaS is favourable.
9. DSO2 conveys 'Access denied' information to the DSO because security decision made by SecaaS is not favourable.
10. The 'Access Denied' information provided by DSO2 is forwarded to the SR by its DSO (DSO1).

Procedure for Requesting a Typical SecaaS Service

Figure 5 is a model showing request and response for a typical security service (e.g. authentication or authorization). The procedure, as schematically shown in Fig. 5, is as follows:

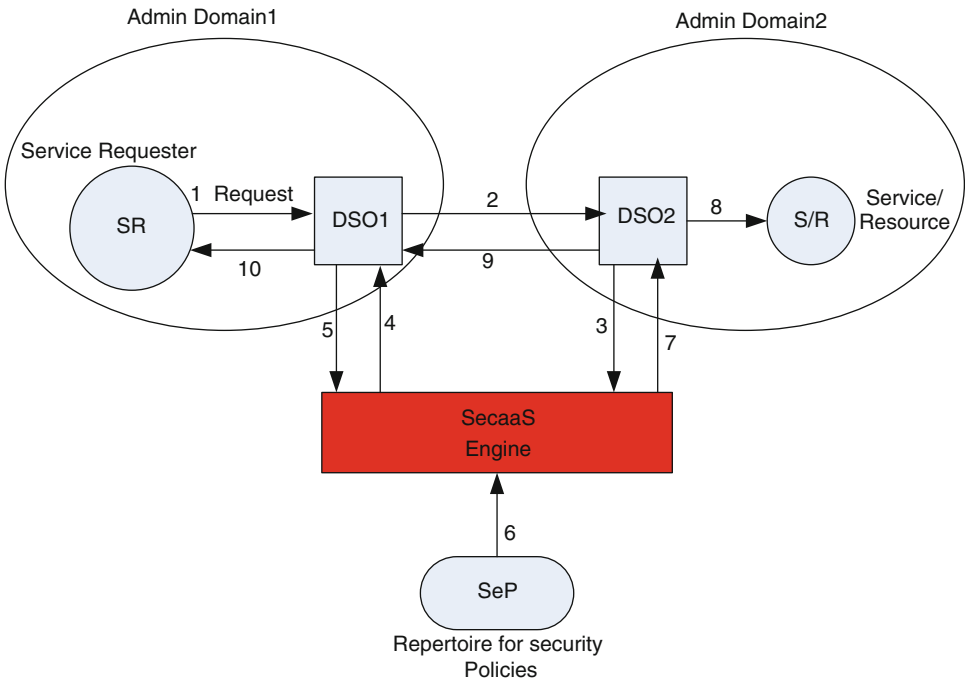

Fig. 4 A schematic diagram showing request for access control services when both the SR and the R/S belong to different Ads

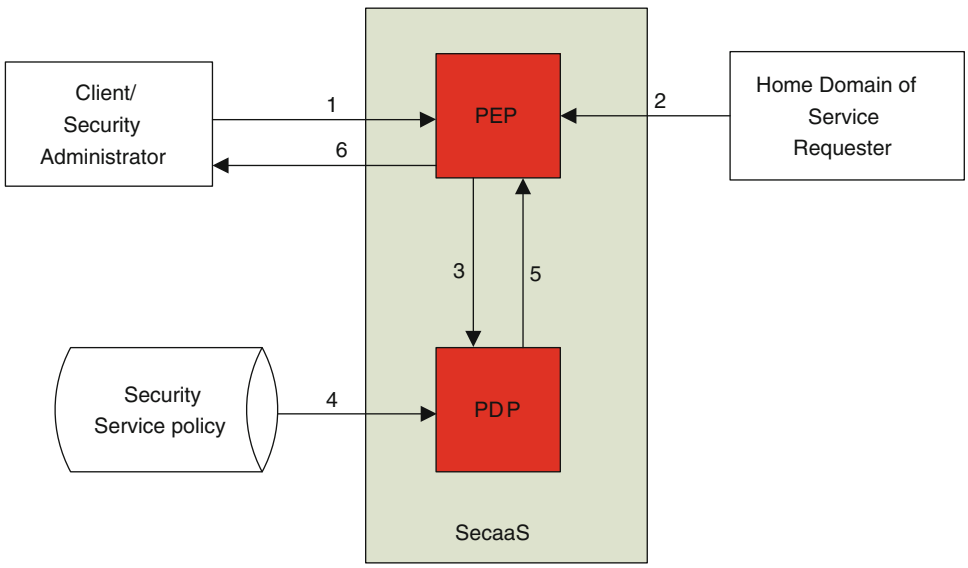

Fig. 5 Request for a typical access control service (e.g. Authorization service)

1. A client request for aZ service from SecaaS (the client, in this case, is the entity requesting for security service; typically it is the DSO). The request is intercepted by PEP of SecaaS.
2. PEP obtains the security token (e.g. aZ privileges) of SR from SR's home domain
3. PEP constructs necessary query and contacts PDP for access decision-making.
4. PDP retrieves appropriate aZ policies associated with the requested R/S and as sent to it by the client and evaluates the request against the policy.
5. PDP sends the result (Grant or Deny access) of aZ policy evaluation to PEP for necessary action. (All that PEP does here is to forward security service decision on the access request to the client).
6. PEP returns access decision (Grant or deny access) to the client (i.e. the DSA that requests for security service)

Note: The security service policy 'database' (a temporary repertoire for security services policies) with SecaaS in Fig. 5 is not static. It varies with each request for security service.

Conclusion and Future Work

A Security model that employs SecaaS approach in addressing security challenges of a grid environment has been presented in this paper. This model is proposed because the solutions offered at addressing the inadequacies of the existing grid infrastructures in satisfying some desirable access control requirements of distributed services result in replication of effort as well as increase in the cost of developing and maintaining applications.

In the SecaaS approach, each atomic access control function/service, such as authentication, authorization, etc. will be provided as re-useable services that can be exposed and then subscribed to by different grid entities. In this security model, each AD in a grid environment will no longer need to have its own domain-specific access control (security) logic built in to it. Whenever a security service is required, the domain subscribes to this service from SecaaS. The potential benefits of employing SecaaS approach at providing security solution for a grid system are as high-lighted in section one.

The next stage in this work is to carry out a prototype implementation of our SecaaS model. We intend to carry out the implementation in .NET environment with the support of WSE 3.0 toolkit, and WSRF. WSE 3.0 has support for web service security specification like ws-security, and ws-secureConversation. Ws-security [13] is the facto standard for security message in the web. It enables one to selectively encrypt or sign different part of single SOAP message for different recipients. Ws-secureConversation [13] was designed to provide secure communication for multiple SOAP messages. WSRF is an OASIS standard, generic and open framework for modelling and accessing stateful resources by using web services. It provides specification for making web service to become stateful [3]. Stateful web services are generally required by grid applications.

The prototype implementation will begin with two ADs, each having 2-5 service providers (SPs) which would provide R/S to other domains. Afterward, more ADs will be added, while also increasing the number of SPs in each domain. R/S will be exposed as (stateful) web services, with each R/S having its own set of security policies as provided by the domain Administrator. Policies associated with a R/S will be stored in a database within its domain in the XACML format.

References

1. Jacob, B., Brown, M., Fukui, K. and Trivedi, N.. Introduction to Grid Computing, IBM Corporation, Ibm.com/redbooks , 2005.
2. Magoules, F., Pan, J., Tan, K. and Kumar, A.. Introduction to Grid computing, London, CRC Press, Chapman and Hall Book, 2009.
3. Sotomayor and Childers. Globus Toolkit 4: programming Java Services, san Francisco, Morgan Kaufmann Publisher, 2006.
4. Singh, S., Singh, K., & Kaur, H. Design and Evaluation of policy-based Authorization Model for large Scale Distributed Systems, IJCSNS International Journal of Computer Science and Network Security, 2009, Vol. 9 No. 11, pg 49-55
5. Jie, W., Arshad, J., Sinnott, R., Townend, P & Lei, Z. A Review of Grid Authentication and Authorization Technologies and Support for Federated Access Control, ACM Computing Survey, vol.43, no 2, Article 12, January 2011.
6. Singh, S. & Bawa, S. A Privacy, Trust and Policy based Authorization Framework for services in Distributed Environment, International Journal of Computer Science, 2007, Vol 2 , No. 2.
7. Zhao, S., Aggarwal, A. & Kent, R.D. PKI-Based Authentication Mechanisms in Grid Systems, International Conference on Networking, Architectures and Storage, IEEE Computer Society, 2007.
8. Singh , S. & Bawa, S. Design of a framework for Handling Security Issues in Grids, IEEE 9th International conference on Information Technology, 2006.
9. Hartman, B., Flinn, D.J., Benznosov, K. & kawamote, S. Mastering Web Services Security, Canada, Wiley Publishing, Inc, 2003.
10. Ekabua, O.O & Adigun M.O. GUISET LogOn: Design and Implementation of GUISET-driven Authorization framework, Proceedings of Cloud computing 2010. The first International Conference on Loud Computing, Grids and Virtualization.
11. Lang, B.O , Foster , I., Siebenlist, F., Ananthakrishnan, R.,& Freeman, T. A Multipolicy Authorization Framework for Grid Security, 2008. http://www.mcs.anl.gov/uploads/cels/papers, Retrieved on 23-05-2011.
12. Squicciarini, A.C., Bhargav-Spantzel, A., Bertino E., & Czeksis, A. B. Auth-SL – A System for the Specification Enforcement of Quality-Based Authentication policies, ICICS, 2007, pg 386-397
13. Bertino, E., Martino, L.D., Paci, F. & Squicciarini, A.C. Security for Web Services and Service- oriented Architectures, London, Springer, 2010.
14. Moses, T. Extensible Access Control Markup Language (XACML), (OASIS Standard, 2005). Available online at http://docs.oasis-open.org/xacml/2.0/; Accessed 12th June, 2012.