Geo-spatial Information Science (Quarterly)
Volume 9, Issue 3
September 2006

Article ID: 1009-5020(2006)03-223-06 **Document code:** A

Dynamically Authorized Role-Based Access Control for Grid Applications

YAO Hanbing HU Heping LU Zhengding LI Ruixuan

ABSTRACT Grid computing is concerned with the sharing and coordinated use of diverse resources in distributed "virtual organizations". The heterogeneous, dynamic and multi-domain nature of these environments makes challenging security issues that demand new technical approaches. Despite the recent advances in access control approaches applicable to Grid computing, there remain issues that impede the development of effective access control models for Grid applications. Among them there are the lack of context-based models for access control, and reliance on identity or capability-based access control schemes. An access control scheme that resolve these issues is presented, and a dynamically authorized role-based access control (D-RBAC) model extending the RBAC with context constraints is proposed. The D-RABC mechanisms dynamically grant permissions to users based on a set of contextual information collected from the system and user's environments, while retaining the advantages of RBAC model. The implementation architecture of D-RBAC for the Grid application is also described.

KEYWORDS Grid security; RBAC; context-based; access control
CLC NUMBER TP393

Introduction

The Grid security infrastructure (GSI) has been accepted as the primary authentication mechanism for the Grid computing. GSI developed as part of the Globus project defines single sign-on algorithms and protocols, cross-domain authentication protocols, and temporary credentials called proxy credentials. GSI is widely used and has been integrated into a number of Grid environments and applications [1], while many research efforts address important aspects of the overall authorization and access control problem in a Grid environment, these efforts focus on relatively static scenarios where access depends on the user's identity (or role) [2-4]. They do not address access control issues for Grid applications where the access capabilities and privileges of a subject not only depend on its identity but also on its security-relevant contextual information, such as time, location, or environmental state available at the time the access requests are made, and incorporate it in its access control decisions. These context parameters capture the dynamically changing access requirements in Grid application, and hence are critical to the effectiveness of the resulting access control scheme. In order that the access control can be effectively exercised in such scenarios with context-based access requirements, the traditional access control models must be extended to make themselves context-based. To this end, we propose a D-RBAC model for Grid applications.

The remainder of the paper is organized as follow. Section 1 presents RBAC model. Section 2 describes our approach, including a brief presentation of security context, and presents a formal

Received on April 29, 2006.
Supported by the National Natural Science Foundation of China (No. 60403027).
YAO Hanbing, Ph. D candidate, College of Computer Science and Technology, Huazhong University of Science and Technology, Guanshan, Wuhan 430074, China.
E-mail: hustyhb@163.com

definition for D-RABC. Section 3 describes D-RABC framework for Grid application. Section 4 concludes this paper.

1 Role based access control

RBAC model was first presented by Sandhu and has recently aroused increasing attention in the security community [5]. As opposed to DAC and MAC model based on a simple subject-object relation, RBAC model is based on three sets of entities called users (U), roles (R), and permissions (P). A user (U) is a human being or an autonomous agent. A role (R) is a job title or a job function in the organization associated with semantics concerning responsibility and authority. The permission (P) is a description of the type of authorized interactions that a subject can have, with one or more objects.

Access control policy is embodied in RABC components such as user-role, role-permission, and role-role relationships. These RBAC components determine whether a particular user is allowed to access to a specific piece of system data. A user can be assigned many roles, and a role can be assigned to many users. The many-to-many assignment relation user-assignment (UA) captures this property. A role can be assigned much permission, and permission can be assigned to many roles. The many-to-many assignment relation permission-assignment (PA) has this property. The formal definition for RBAC is as follows.

1) U, R, P, S which are, respectively, the sets of users, roles, permissions, sessions.

2) $UA \subseteq U \times R$, which is a many-to-many user-assignment relation assigning a user to roles.

3) $PA \subseteq P \times R$, which is a many-to-many, permission-assignment relation assigning permissions to roles.

4) $RH \subseteq P \times R$ is a partial order on R called role hierarchy.

5) user: $S \rightarrow U$, is a function mapping each session s_i to the single user $user(s_i)$ and is constant for the session's lifetime.

6) roles: $S \rightarrow 2^R$ is a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)\}$ $[(user(s_i), r' \in UA]\}$ so that session s_i has the permissions $U_r \in roles(s_i)\{p \mid (\exists r'' \leq r)[(p, r'' \in PA]\}$

Sandhu defines a comprehensive framework for RBAC models which are characterized as follows.

1) $RBAC_0$: the basic model where users are associated with roles and roles are associated with permissions.

2) $RBAC_1$: $RBAC_0$ with role hierarchies.

3) $RBAC_2$: $RBAC_1$ with constraints on user/role, role/role, and/or role/permission associations.

RBAC allows to express and enforce enterprise-specific security policies and which simplifies the administration of access rights. Users can make members of roles as determined by their responsibility and qualification and can be easily reassigned from one role to another without modifying the underlying access structure. Roles can be granted with new permissions, or permissions can be revoked from roles as needed. RBAC can be used by the security administrator to enforce the principle of least privilege as well as static, dynamic, and operational policies of separation of duties.

Recently RBAC has been found to be the most attractive solution for providing security in a distributed computing infrastructure [6]. Although the RBAC models vary from very simple to pretty complex, they all share the same basic structure of subject, role and privilege. Other factors, such as relationship, time and location, which may be part of an access decision, are not considered in these models. The D-RBAC model presented in this paper extends RBAC to provide context-aware access control mechanisms for Grid applications.

2 D-RBAC model

The D-RBAC mechanisms dynamically grant and adapt permissions to users based on a set of contextual information collected from the system and user's environments. The D-RBAC model

extends the RBAC in context and content-based constraints, while retaining its advantages (i.e. ability to define and manage complex security policies). RBAC addresses many other issues such as role activation, revocation, role hierarchies and separation of duty constraints. These issues apply to D-RBAC as well.

2.1 Context-based security

As its name suggests, context-based security is all about considering "context" explicitly in the specification of access control models [7-9]. Fig.1 illustrates the idea behind context-based security in the grid application. The grid environment is initially controlled with a specific configuration of the security policy in an initial context. This context is continually changing in request to triggers (dynamic changes in the environment). The security policy must then adapt itself to the new context.

By a security policy, we mean a specification that expresses clearly and concisely what access request are authorized and what are those that are denied for each type of user in each situation. Formally, a situation is what we call a security context as. Context-based security adapts itself to cope with the new types of security problems introduced by the heterogeneous, dynamic and multi-domain nature in grid environments.

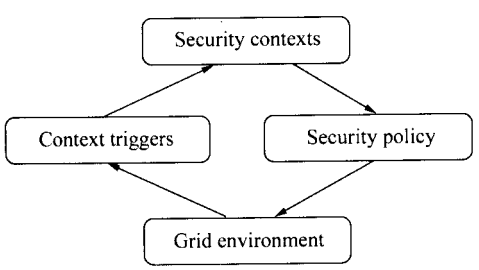

Fig.1 Context-based security in Grid application

2.2 Formal definition for security context

This section defines the set of specifications needed to define D-RBAC for context-based access control in Grid applications. We provide the formal definition of security context below. In order to formalize the security context, we introduce a type system to allow specifying domains of legal values for various context parameters. The D-RBAC model relies on the components we define below.

Definition 1 Context parameter (CP): a context parameter is represented by a data structure p having the following fields: name $\in$CN, type $\in$CT, and a context function getValue(). The CN is a set of the possible names of context parameters, and the CT is a set of types of context parameters, and the context function of getValue() is a mechanism to obtain runtime values for specific context parameter. CP represents a certain property of the environment whose actual value might change dynamically (like time, date, or session-data, for example). For example, the set CN may be defined as CN = {time, location, duration, system_load}, with the corresponding set CT defined as: CT = {time, string, long, integer}.

Context parameter is separated from the main business logic of target applications. Since every context type definition is independent of the specification of the access rules, any change of them has no effect on other parts of the system.

Definition 2 Context set (CS): a context set CS consists of n context parameters $\{CT_1, CT_2, \cdots, CT_n\}$, $n \geq 0$, for any CT_i, CT_j, with $i \neq j$ and $1 \leq i, j \leq n$, we have that $CT_i.name \neq CT_j.name$ (i.e. the parameter names must be distinct). By analyzing the grid application security requirements, application designers determine which context types will be used to specify access policy. Although the context set is determined before the application implementation, system administrators can dynamically add new ones when there are needs.

2.3 Formal definition for D-RBAC

On the basis of the formalization of the RBAC model, we present a precise description of D-RBAC model including security-relevant contextual information. Both role hierarchies and separation of duty in RBAC are meaningful in the D-RBAC, though they are omitted here in our description. We only consider flat user and security-relevant contextual information. This

formalization can be extended to hierarchies and constraints similar to the $RBAC_1$ and $RBAC_2$ models. An overview of the D-RBAC is shown in Fig 2. We keep USERS, ROLES, OBS, OPS, PRMS and SESSIONS in the RBAC.

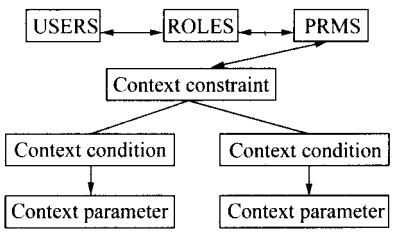

Fig. 2 D-RBAC

Definition 3 Context condition (CN): CN = < CT > < OP > < Value >, CT ∈CS, OP is a standard comparison and logical operator, Value is a specific value, and the type of Value is CP.type.

A context condition is a predicate (a Boolean function) that compares the current value of a context parameter with a predefined constant. The corresponding comparison operator must be such an operator that it is defined for the respective domain. All variables must be ground before evaluation. Therefore each context parameter is replaced with a constant value by using the according context function prior to the evaluation of the respective condition. Examples for context conditions can be CN_1: Date() <"2006-01-01", CN_2: age(subject) > 18.

Definition 4 Context constraints (CC): CC = $CL_1 \cup CL_2 \cdots \cup CL_n$, CL = $CN_1 \cap CN_2 \cdots \cap CN_n$. CC means also context condition. Based on this format, our access control schema is capable of specifying any complex context related constraint to describe all kinds of security requirements. System administrators can dynamically adapt context constraint.

Context constraints are conditions with which an object must be satisfied in order the user's attempt to perform an operation succeeds. These conditions involve security-relevant parameters of the attempted operation. This may include information gleaned from environment (such as the time of day, or whether it is a holiday), or state contained in the target object itself. These constraints are distinct from those defined in the base RBAC model, which constrain role definitions in order to avoid conflicting roles, promote separation of duties, etc. Systems such as [7] allow constraints, in the form of environment roles that are purely dependent on external properties rather than the properties of the objects or subjects, are involved in the operation. The Role Object Model defines a role as a set of policies. Constraints involving properties of the objects are used to limit the applicability of those policies over object instances [10].

Definition 5 D-RBAC: D-RBAC = {USERS, ROLES, OBS, OPS, PRMS, SESSIONS, CC}. The USERS, ROLES, OBS, OPS, PRMS and SESSIONS are defined in RBAC, the CC is context constraint.

Definition 6 Access policy (AP): we define an access policy as a triple, AP = (R, P, C), R ∈ROLES, P ∈PRMS, C ∈CC. If C is empty then this policy reverts to simple RBAC.

Definition 7 Access request (AR): we define access request as a triple, AR = (R′, P′, RC), R′∈ROLES, P′∈PRMS, RC(runtime context) is a set of values for every context type in the context set. That is, RC = {CT_1.getvalue(), CT_2.getvalue(), …, CT_n.getvalue()}, {CT_1, CT_2, …, CT_n} is the context set (CS) of the grid application.

An access request is granted only if there exists an access policy AP (R, P, C), so that R′= R, P′= P, and C evaluates the true under RC (that is, when all CP_i in context constraint C are replaced with their values in RC, then the resulted Boolean expression is true).

2.4 Dynamic context evaluation algorithms

We can design the basic algorithm to determine whether an access request is authorized or not based upon the context parameter in our model. The algorithm is shown in Fig. 3.

The application passes an AR to the algorithm RequestPermission, and receives a Boolean value in return-indicating whether the attempted operation should be allowed or not. The ar contains the caller's roles and permissions and context

constraints. The access control system first checks whether the application's access policy contain the user's access request, then the context constraints are populated by plugging in values from the application's runtime environment. For each context condition, it is examined if the corresponding runtime value can be captured by an actual context function of the context parameter. If, however, no appropriate context function is available, one can implement a new context function in order to enforce the corresponding context condition without any effect on other parts of the system.

```
Algorithm 1: RequestPermission (AccessRequest ar)
CPS = {} //initialize candidate access policy set
for each AP in PS//PS are policy set
  if (ar.R′ ∈AP.R) and (ar.P′= AP.P)
    put AP into CPS
  end if
end for
result = false
for each AP in CPS
    if (EvaluateContexts(AP.C) is true)
     result = true
     break
    else
     result = false
  end if
end for
return result
```

```
Algorithm 2: EvaluateContexts(Constraint rc)
for each CL in rc
  for each CN in CL
    if (<CP.getvalue() > <OP> <VALUE> = false)
      //CP.getvalue() get CPS runtime value
      //OP is specific operrator of CN
     CL = false
     break
     end if
  end for
  if (CL = true)
    return true
  else
    continue
  end if
end for
return false
```

Fig. 3 Algorithms for D-RBAC

3 D-RBAC framework for Grid applications

A prototype of the D-RBAC model has been implemented as part of our lab's Grid system on the top of OGSI. It is a Grid-based computational collaboration that enables scientists and engineers over all the world to collaboratively access, monitor, and control distributed applications, services, resources and data on the Grid using grid portal. Key components of D-RBAC framework are listed as follows.

Grid portals: providing users with pervasive and collaborative access to Grid applications, services and resources. Using these portals, users can discover and allocate resources, configure and launch applications and services, and monitor, interact with, and steering their execution. The Grid portals include authentication module and global authority service module [10].

User context agent: capturing all security-relevant information about a particular user.

Object context agent: capturing all security-relevant information about the target object.

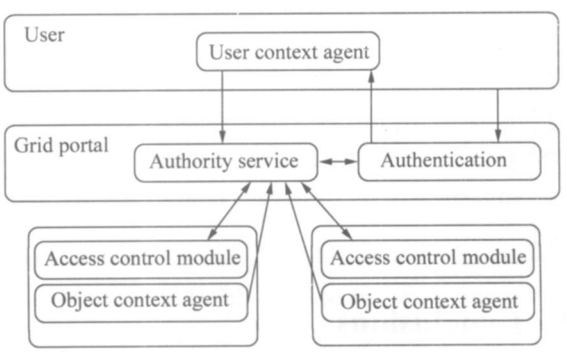

Fig. 4 D-RBAC framework for Grid application

An overview of the D-RBAC for Grid applications is shown in Fig 4. The D-RBAC model ensures the users access, monitoring and steering Grid resources/applications/services only if they have appropriate privileges and capabilities. As the Grid environment is dynamic, so it requires dynamic context aware access management. Note that authentication services are provided by GSI.

In our implementation, users entering the Grid application using the portal are assigned a set of roles when they log in. A user context agent is

then locally set up for each user, which dynamically adjusts the user context. Similarly, the object context agents are set up at the application (or service/resource) for each role that will access it. The object context agents similarly adjust the object context.

Assuming that the following access request is submitted for the evaluation of the Grid application:

< R = "guest", P = "view", C = { p_1 {time, Time}, p_2 {location, String}, p_3 {duration, Long}, p_4 {system_load, Integer}} >.

The context recorded at the time of access request is captured by context agent, and provided to the system as part of the request. Now, assuming that the following AP is applicable to the permission P.

< R = "guest", P = "view", C = CC >

$CC = CL_1 \cap CL_2 \cap CL_3 \cap CL_4$

CL_1: {time > 8:00} AND {time < 18:00)

CL_2: {location = "admin1"} OR {location = "admin2"}

CL_3: {duration $\leq$600 s}

CL_4: {system_load ! = "high">

On the basis of this information, the system would return authorization decision for this access request. The available contextual information indicates that the access conditions are satisfied.

4 Conclusions

We described a D-RBAC infrastructure that extends the traditional RBAC model to gain many advantages from its context-based capability. Our research motivation comes from the complicated access control requirements in Grid application. Traditional RBAC is not able to specify a sufficiently fine-grained authorization policy or specify constraints that should be applied to an access policy. Our new access control infrastructure is dynamic and has following advantages.

1) The D-RBAC model extends traditional RBAC by associating access permissions with context-related constraints. Every constraint is evaluated dynamically against the current context of the access request. Therefore, the model is capable of making authorization decisions based upon context information in addition to roles.

2) Our context-based access control is applied dynamically. At design time, administrators have great flexibility to specify complex context-based authorization policies. At run-time, our authority service can enforce any context-based policy automatically because it is not statically bound to any application.

3) Context information is separated from the main business logic of target applications. Since every context type definition is independent of the specification of the access rules, any change in them has no effect on other parts of the system. Thus our security infrastructure is flexible and permits high extensibility. Although context constraints can be modeled and used in a straightforward manner, they may potentially add a great deal of complexity to access control policies. On the other hand, they add much flexibility and expressiveness, and allow for the definition of fine-grained access control policies as they are often needed in real-world applications. We intend to report the detailed results of our on-going implementation efforts in some future work. We also plan to explore the interplay of contextual conditions in the presence of separation of duty constraints. Separation of duty principles are a type of access control policy which require two or more users being responsible for the completion of a business process. By distributing the responsibility of a business process between numerous users, there are fewer opportunities for one user to commit a fraudulent act without being discovered. It is critical to ensure that the access to grid resources based on context constraints do not violate any separation of duty constraints.

(Continued on Page 233)

every node can hear the neighbor's radio without being detected. When two or more malicious nodes construct one or more wormholes, they can destroy the entire Network by disrupting the routing protocol, especially to OLSR protocols.

In this paper we introduced a trust model to evaluate the trustiness of "a node is the neighbor" in OLSR protocol. From the trustiness calculating, the node can get the right route instead of choosing the route caused by wormhole attack. This scheme can run with no need for network synchronization and GPS devices. But the scheme is based on trust evaluation, which predicts the future events by collecting the past events, so the trust evaluated by the node lags behind the attacks.

In future work, we will work on how to secure the trustiness message transmission and how to get the recommended path in trust graph. We also take the node's mobility into consideration, because when the network topology changing fast, the route will change fast, which means the trust model should keep track with it.

REFERENCES

[1] Clausen T, Jacquet P (2003) Optimized link state routing protocol (OLSR) [OL]. http://www.ietf.org/rfc/rfc3626.txt

[2] Hu Yihchun, Perrig A (2004) A survey of secure wireless ad hoc routing[J]. *IEEE Security and Privacy*, 2(3):28-39

[3] Hu Yihchun, Perrig A, Johnson D B (2003) Packet leashes: a defense against wormhole attacks in wireless Networks[C]. The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, CA, USA

[4] Hu Yihchun, Perrig A, Johnson D B (2003) Rushing attacks and defense in wireless ad hoc Network routing protocols[C]. The Workshop on Wireless Security, San Diego, CA, USA

[5] Hu Lingxuan, Evans D (2004) Using directional antennas to prevent wormhole attacks[C]. Network and Distributed System Security Symposium, San Diego, California

[6] Marsh S (1999) Formalizing trust as a computational concept[D]. U. K.: Department of Mathematics and Computer Science, University of Stirling

(Continue from Page 228)

REFERENCES

[1] Foster I, Kesselman C, Tsudik G, et al. (1998) A security architecture for computational grids[C]. The 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA

[2] Johnston W, Mudumbai S, Thompson M (1998) Authorization and attribute certificates for widely distributed access control[C]. The Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, California, USA

[3] Pearlman L, Welch V, Foster I, et al. (2002) A community authorization service for group collaboration[C]. The 3rd International Workshop on Policies for Distributed Systems and Networks, Monterey

[4] Alfieri R, Cecchini R, Ciaschini V, et al. (2004) VOMS, an authorization system for virtual organizations[C]. The First European Across Grids Conference, Santiago de Compostela

[5] Sandhu R S, Coyne E J, Feinstein H L, et al. (1996) Role-based access control models[J]. *IEEE Computer*, 29(2):38-47

[6] Moyer M J, Ahamad M (2001) Generalized role-based access control[C]. International Conference on Distributed Computing Systems. Mesa, AZ

[7] Emil L, Morris S (1997) Policy based role object model[C]. The International Enterprise Distributed Object Computing Workshop, Australia

[8] Patrick M (2003) On context in authorization policy [C]. ACM Symposium on Access Control Models and Technologies, Villa Gallia, Como, Italy

[9] Kumar A, Karnik N, Chafle G (2002) Context sensitivity in role-based access control [J]. *Operating Systems Review*, 36(3):53-66

[10] Hu Heping, Yao Hanbing (2005) A scheme for authentication and authorization in a grid application [C]. The 19th International Conference on Advanced Information Networking and Applications, Taipei, Taiwan