Dynamic Context-aware Access Control for Grid Applications*

Guangsen Zhang, Manish Parashar
The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering
Rutgers University
{gszhang,parashar}@caip.rutgers.edu

Abstract

The emerging Grid infrastructure presents many challenges due to its inherent heterogeneity, multi-domain characteristic, and highly dynamic nature. One critical challenge is providing authentication, authorization and access control guarantees. In this paper, we present the SESAME dynamic context-aware access control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underling dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC). We also present a prototype implementation of SESAME and DRBAC with the Discover computational collaboratory and an experimental evaluation of its overheads.

Keywords: Grid security, authorization and access control, context-aware, pervasive applications, Grid computing.

1 Introduction

Grid computing is rapidly emerging as the dominant paradigm of wide area distributed computing [1]. It's primary objective is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources.

The Grid community and the Global Grid Forum [15] are investing considerable effort in developing and deploying standard architectures and protocols that enable seamless and secure discovery, access to, and interactions among resources, services, and applications. This potential for seamless aggregation, integration, and interactions has made it possible to conceive a new generation of Grid applications that are based on ad hoc, symbiotic and opportunistic interactions, where users, application components, Grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) interact as peers. However, realizing such a pervasive Grid infrastructure presents many challenges due to its inherent heterogeneity, multi-domain characteristic, and highly dynamic nature. One critical challenge is providing authentication, authorization and access control guarantees.

The Grid Security Infrastructure(GSI) [5] has been accepted as the primary authentication mechanism for the Grid. Developed as part of the Globus project [16], GSI 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.

However, the authorization and access control challenges are not fully addressed by existing approaches. The Akenti [4] access control system enables multiple owners and administrators to define fine-grained usage policies in a widely distributed system. The Akenti policy engine then gathers use-conditions certificate defined by the resource owners and attribute certificates from the various stake holders, and grants access to a resource by matching of these two certificates. In the Community Authorization Service (CAS) [3], resource providers grant access to a community accounts as a whole. The CAS server is designed to maintain authorization information for all entities in the community. It keeps track of fine-grained access control information and grants restricted GSI proxy certificates (PCs) to community members. M. Lorch et al [6] propose a fine grained authorization services to support ad-hoc collaborations using attribute certificates. Similarly, L. Ramakrishnan et al [7] present an authorization infrastructure for component-based Grid applications by providing authorization at the component interface.

While these research efforts listed above do address im-

*The research presented in this paper is supported in part by NSF via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR), and by DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

portant aspects of the overall authorization and access control problem in a Grid environment, these efforts focus on relatively static scenarios where access depends on identity of the subject. They do not address access control issues for pervasive Grid applications where the access capabilities and privileges of a subject not only depend on its identity but also on its current context (i.e. current time, location, system resources, network state, etc.) and state. For example, consider a user accessing a remote resource or a data archive using a pervasive portal on her PDA. In such an application, the user's access privileges depend on who she is, where she is (in a secure or insecure environment), her context (current connectivity, current load), the state of the resource or data archive she is accessing, etc. Furthermore, her privileges will change as her context changes - for example, if she moves from a secure wireless link to an insecure one. Similarly, when a Grid service interacts with another service on the Grid, the access privileges of the service will also depend on the credential of the service as well as the context and state of the service, which are dynamic.

In this paper, we present the SESAME [1] dynamic context-aware access control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underling dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC) [2, 8], while retaining its advantages (i.e. ability to define and manage complex security policies). The model dynamically adjusts *Role Assignments* and *Permission Assignments* based on context information. In DRBAC, each subject is assigned a role subset from the entire role set by the authority service. Similarly, each object has permission subsets for each role that will access it. During a secure interaction, state machines are maintained by delegated access control agents at the subject (*Role State Machine*) to navigate the role subset, and the object (*Permission State Machine*) to navigate the permission subset for each active role. These state machines navigate the role/permission subsets to react to changes in context and define the currently active role at the subject and its assigned permissions at the object.

A prototype of SESAME and the DRBAC model has been implemented as part of the Discover [11, 12] computational collaboratory. Discover enables geographically distributed scientists and engineers to collaboratively access, monitor and control applications, services, resources and data on the Grid using pervasive portals. The feasibility, performance and overheads of SESAME are experimentally evaluated.

The rest of this paper is organized as follows. Section 2 presents the SESAME dynamic access control model and describes its operation. Section 3 describes the prototype implementation within the Discover collaboratory. Section 4 presents an experimental evaluation. Section 5 presents a summary and conclusions.

[1] Scalable, Environment Sensitive Access Management Engine

2 Dynamic Role-based Access Control

As mentioned above, a key requirement for pervasive Grid applications is the support for dynamic, seamless and secure interactions between the participating entities, i.e. components, services, applications, data, instruments, resources and users. Guaranteeing interaction security requires a fine-grained access control mechanism. Furthermore, in the highly dynamic and heterogeneous Grid environment, the access privileges of an entity depend on its credential, context and current state, which are dynamic. In this section, we present the SESAME Dynamic Role Based Access Control model(DRBAC) to address these requirements. The traditional Role Base Access Control(RBAC) model is first discussed. The DRBAC model and its operation are then described in detail.

2.1 RBAC

Role based access control (RBAC) is an alternative to traditional discretionary (DAC) and mandatory access control (MAC). In RBAC, users are assigned roles and roles are assigned permissions. A principle motivation behind RBAC is the ability to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure. As user/role associations change more frequently then role/permission associations, in most organizations, RBAC results in reduced administrative costs as compared to associating users directly with permissions. It can be shown that the cost of administrating RBAC is proportional to U+P while the cost of associating users directly with permissions is proportional to U*P, where U is the number of individuals in a role and P is the number of permissions required by the role [8]. Sandhu [2] defines a comprehensive framework for RBAC models which are characterized as follows:

- $RBAC_0$: the basic model where users are associated with roles and roles are associated with permissions.
- $RBAC_1$: $RBAC_0$ with role hierarchies.
- $RBAC_2$: $RBAC_1$ with constraints on user/role, role/role, and/or role/permission associations.

Recently RBAC has been found to be the most attractive solution for providing security in a distributed computing infrastructure [8]. 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 SESAME DRBAC model presented in this paper extends RBAC to provide context-aware access control mechanisms for dynamic and pervasive Grid applications.

2.2 Dynamic Role-based Access Control Model

The formalization of the DRBAC model is based on the RBAC model presented in [9]. The DRBAC model is illustrated in Figure 1. It has the following components:

- USERS. A user is an entity whose access is being controlled. USERS represents a set of users.
- ROLES. A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. ROLES represents a set of roles.
- PERMS. A permission is an approval to access one or more DRBAC protected resources. PERMS represents a set of permissions.
- ENVS. ENVS represents the set of context information for the system. We use an authorized "context agent" to collect context information in our system.
- SESSIONS. A session is a set of interactions between subjects and objects. SESSIONS represents a set of sessions.
- UA. UA is the mapping that assigns a role to a user. In a session, each user is assigned a set of roles and the context information is used to determine the active role among these. The user accesses the resource using this active role.
- PA. PA is the mapping that assign permissions to a role. Every role which has privileges to access the resource is assigned a set of permissions and the context information is used to determine the active permissions for the roles.

In the DRBAC model, a Central Authority (CA) maintains the overall role hierarchy for each domain. When the subject logs into the system, based on her credential and capability, a subset of the role hierarchy is assigned to her for the session. The CA then sets up and delegates (using GSI) a local context agent for the subject. This agent monitors the context for the subject (using services provided by the Grid middleware) and dynamically adapts the active role. Similarly every subject maintains a set of permission hierarchies for each potential role that will access the resource. A delegated local context agent at the subject resource will

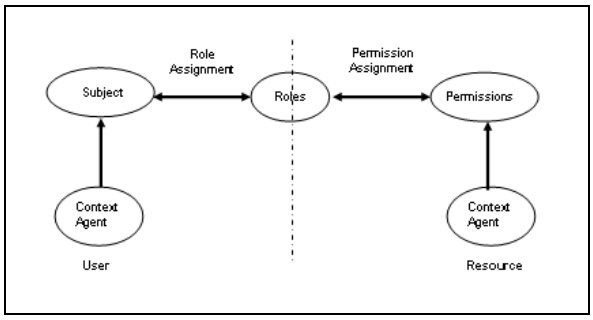

Figure 1. The dynamic access control model

use environment and state information to dynamically adjust the permissions for each role. We formally define the DRBAC model as follows:

- *USERS, ROLES, PERMS, ENVS and SESSIONS (users, roles, permissions, environments and sessions, respectively).*
- ACT_ROLE *and* $ACT\_PERMISSION$ *(active role and active permission respectively).*
- *UA*$\subseteq$*USERS*$\times$*ROLES, a many-to-many mapping user-to-role assignment relation.*
- *PA*$\subseteq$*PERMS*$\times$*ROLES , a many-to-many mapping permission-to-role assignment relation.*
- *Assigned_roles(u:USERS, e:ENVS)* $\rightarrow 2^{ROLES}$*, the mapping of user u onto a set of roles.*
- *Assigned_permissions(r:ROLES, e:ENVS)*$\rightarrow 2^{PERMS}$*, the mapping of role r onto a set of permissions.*
- *User_sessions(u:USERS)* $\rightarrow 2^{SESSIONS}$*, the mapping of user u onto a set of sessions.*
- *Session_roles(s:SESSIONS)* $\rightarrow 2^{ROLESS}$*, the mapping of session s onto a set of roles. Formally: session_roles(*s_i*)* $\subseteq$ $\{r \in$*ROLES*$|$*(session_roles(*s_i*), r)*$\in$*UA*$\}$
- *RH* $\subseteq$ *ROLES*$\times$*ROLES is a partial order on ROLES called the inheritance relation, written as* $\geq$ *, where* $r_1 \geq r_2$ *only if all permissions of* r_2 *are also permissions of* r_1*, and all users of* r_1 *are also users of* r_2*.*
- *PH* $\subseteq$ *PERMS*$\times$ *PERMS is a partial order on PERMS called the inheritance relation, written as* $\geq$ *, where* $p_1 \geq p_2$ *only if all roles of* p_1 *are also roles of* p_2*.*

In the formal definitions above, UA (user assignment) defines the relationship among roles, users and environments; PA (permission assignment) defines the relationship among

permissions, roles and environments. RH (role hierarchy) and PH (permission hierarchy) define the inheritance relationship among roles and permissions respectively. The following section explains the operation of our model in detail.

2.3 DRBAC Operation

In the DRBAC model, we assign each user a role subset from the entire role set. Similarly each resource will assign a permission subset from the entire permission set to each role that has privileges to access the resource. Figure 2 shows the relationship between the role hierarchy maintained at the Central Authority (CA) and the subset of this hierarchy assigned to a particular user.

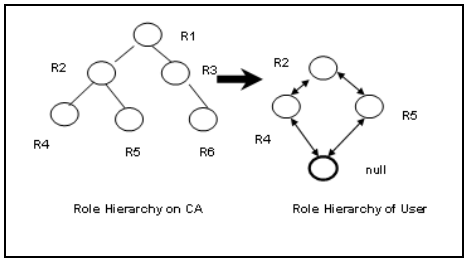

Figure 2. Role hierarchy state machine

We use state machines at the subject (Role State Machine) to maintain the role subset for a user, and at the object (Permission State Machine) to maintain the permission subset for each role. A state machine consists of state variables (a role or permission) that encode state, and events that transform its state. The delegated local context agent uses middleware services to monitor context and generates events to trigger a transition of the state machine when necessary.

A permission hierarchy is shown in the Figure 3. Note that the null permission signifies no access privileges. A transition is defined as *T(Initial State, Destination State)*. So *T(P1, P2)* represents the transition from P1 to P2 and *T(P2, P1)* represents the transition from P2 to P1. In this example, P2 is the current active permission. Role transitions in the Role State Machine are similarly defined.

Key concerns in the implementation of the proposed state machine based access control mechanism include its performance overheads and the reliability and security of the context information. In a typical organization, the number of roles and permissions is relatively small, no more than 20. As a result, with the increasing computational capability of systems, maintaining the state machine will have little if any impact on performance. Also, there are a number of research and commercial efforts [14] developing context toolkits that can provide reliable and secure context services.

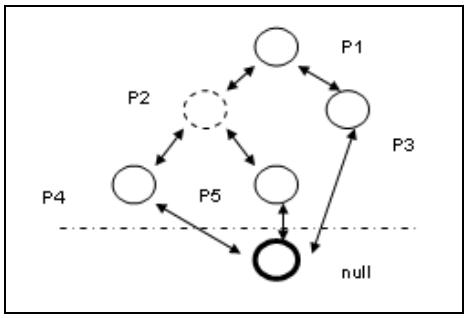

Figure 3. Permission hierarchy state machine

3 SESAME/DRBAC Prototype Implementation

A prototype of SESAME and the DRBAC model has been implemented as part of the Discover [11, 10] computational collaboratory. Discover is a Grid-based computational collaboratory that enables geographically distributed scientists and engineers to collaboratively access, monitor, and control distributed applications, services, resources and data on the Grid using pervasive portal. Key components of the Discover collaboratory include:

- **Discover Collaborative Portals** [11] that provide 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 steer their execution.
- **Discover Middleware Substrate** [12, 10] that enables global collaborative access to multiple, geographically distributed instances of the Discover computational collaboratory, and provides interoperability between Discover and external Grid services such as those provided by Globus [16].
- **DIOS Interactive Object Framework (DIOS)** [13] that enables the runtime monitoring, interaction and computational steering of Grid applications and services. DIOS enables application objects to be enhanced with sensors and actuators so that they can be interrogated and controlled.

An overview of the integration of SESAME and DRBAC with Discover is presented in Figure 4. SESAME ensures the users can access, monitor and steer Grid resources/applications/services only if they have appropriate privileges and capabilities. As Discover portals are pervasive and the Grid environment is dynamic, this requires dynamic context aware access management. Note that authen-

tication services are provided by GSI [5] in our prototype implementation.

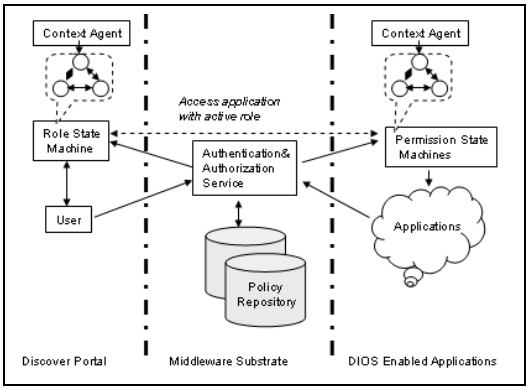

Figure 4. Dynamic access control in discover

In our implementation, users entering the Discover collaboratory using the portal are assigned a set of roles when they log in. A *Role State Machine* is then locally set up for each user, which dynamically adjusts the active role based on events from the local context agent. Similarly, the *Permission State Machines* are set up at the application (or service/resource) for each role that will access it. The *Permission State Machines* similarly adjust the active permissions based on events from the local context agent. The context agents are authorized by the central authority using GSI delegation mechanisms. The access control policy is stored in the policy repository, which is maintained by an *Authentication & Authorization Service* within *Discover Middleware Substrate*. Polices are specified in XML and define role/permission assignments and transitions as illustrated in Figure 5. Policies defined for our implementation include *UserPolicy*, *RoleHierarchyPolicy*, *RoleAssignmentPolicy*, *PermissionAssignmentPolicy*, *EventPolicy*, *RoleTransitionPolicy* and *PermissionTransitionPolicy*.

```
<ROLE_TRANSITION>
    <POLICY>
        <SUBJECTID>gszhang</SUBJECTID>
        <BEGIN_ROLE>Super User</BEGIN_ROLE>
        <EVENT>Unsecure Link</EVENT>
        <END_ROLE>General User</END_ROLE>
    </POLICY>

</ROLE_TRANSITION>
```

Figure 5. Sample RoleTransition policy in XML

In our prototype implementation, we assume that a security administrator will guarantee the correctness of a policy for a object or subject - i.e. SESAME sets up the *Role State Machines* and *Permission State Machines* without considering checking them for errors or conflicts. There are no inherent constraints on the number of roles and permissions, or on the relationships betweens the roles or permissions. To illustrate our implementation, consider a simple example with a single user with three roles and a Grid resource with three permissions, as shown in Table 1 and Table 2 respectively. The role and permission hierarchies for this example are shown in Figure 6.

Table 1. Permission assignments for the example.

Role	Permissions
Super User	P_1, P_2, P_3
Basic User	P_2, P_3
Guest	P_3

Table 2. Permission definition for the example.

Permission	Privileges
P_1	Steer Object, View Object, Basic
P_2	View Object, Basic
P_3	Basic

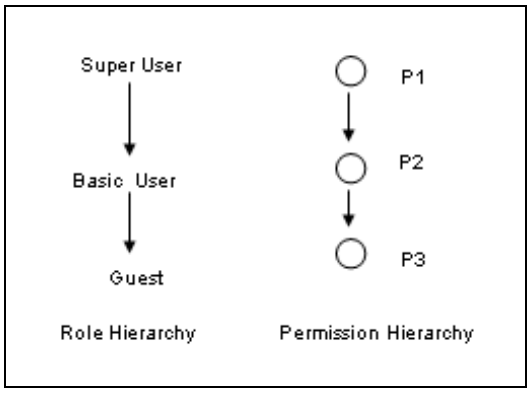

Figure 6. Role and permission hierarchies for the example.

We consider two types of context information in our implementation: (1) Object context such as a user's location, time, local resource state and link state, and (2) Subject context, such as the current load, availability, connectivity for a

resource. Context agents build on existing Grid middleware services. For example object context can be collected using the Context Toolkit [14] and subject context can be obtained using NWS [17].

3.1 SESAME/DRBAC Operation

The operation of the prototype is illustrated using a set of simple scenarios. These scenarios, although somewhat contrived, demonstrate the effectiveness and utility of the DRBAC model for Grid applications. For each of these scenarios, consider a user (say *N*) equipped with a mobile devices such as a PDA, and involved in collaboration scientific investigation using Discover. Assume that the user's environment is part of the pervasive Grid environment with appropriate middleware services.

Assume that user *N* logs into the system using her PDA. Based on her credentials, the *Authentication & Authorization service* assigns her a set of roles. The *Authority Service* also sets up an access control agent on her PDA, which maintains the role state machine. A DRBAC policy defined to select an appropriate role based on the level of security of her wireless connection, i.e. her active role is *Super User* while the network is secure (e.g. in her laboratory or office) and is *Basic User* if it is insecure. The corresponding *EventPolicy* and *RoleTransitionPolicy* may be defined as follow:

- *EventPolicy* - Generate event *insecure* when *N*'s link has no encryption.

- *RoleTransitionPolicy* - Transit role from *Super User* to *Basic User* when event *insecure* is generated.

A corresponding permission state machine is maintained on the application side as shown in Figure 7. As seen in the figure each role has its own permission state machine. The dashed circle represents the current active permission for each role. A DRBAC policy is defined so that the active permission of the role *Super User* is P_1 while load is low and P_2 when the system load increases above some threshold, as there is a possibility that the application may get corrupted. The corresponding *EventPolicy* and *PermissionTransitionPolicy* may be defined as follow:

- *EventPolicy* - Generate event *highload* when load increases above *Threshold*.

- *PermissionTransitionPolicy* - Transit permission from P_1 to P_2 when event *highload* is generated.

Based on the policies defined above, the following scenarios illustrate the operation of the SESAME DRBAC model.

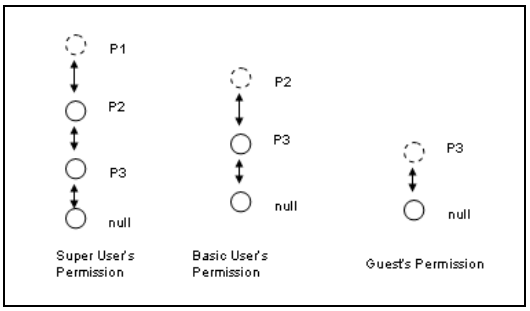

Figure 7. Permission hierarchy for the application

- When user *N* moves out of her laboratory, the context agent will detect (using middleware context services) that the wireless network no longer has the level of encryption required and will generate the *insecure* event. This event will trigger a transition in the role state machine and downgrade her active role to *Basic user*. As a result of this transition, *N* will not be able to control and steer applications as she did while in her laboratory. When she reaches her office where the network is once again secure, the agent will detect this and will once again make *Super User* the active role.

- While in her office, *N*'s active role is *Super User* and she can monitor, interact with and steer applications under normal circumstances (load at the application server is low). However if the load on the application server increases as more users join the session, the local agent generates the *highload* event, which triggers a transition in the permission state machine and change from P_1 to P_2. As a result *Super User* will no longer be able to steer the application.

A screen dump from the *Discover Portal* during these scenarios is illustrated in Figure 8. As shown in this figure, due to the transitions, the portal displays "You don't have the permission to access ....". Note that for these scenarios and the experiments presented in the following section, context information was simulated.

In our current implementation of the DRBAC model, the active role of the user and the active permission of the role change independently. As a result, it is possible that even though the active role of user has been changed to match the current context, the user has certain permission(s) based on the previous role. We are currently addressing this potential consistency issue.

4 Experimental Evaluation

We use the prototype implementation of SESAME in Discover to measure the overheads of the DRBAC model.

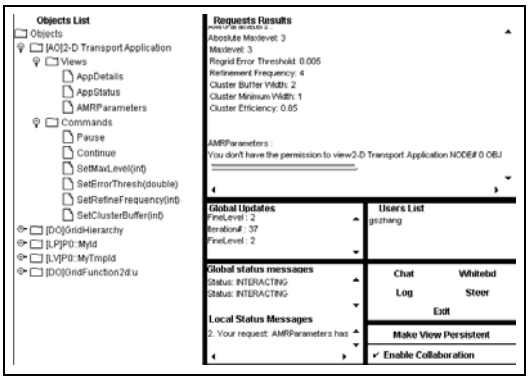

Figure 8. Dynamic access control in discover

The experiments were conducted on two PC using PII-200MHZ processors, running Windows NT 4.0, and one PC using PIII-500MHZ processor, running RedHat Linux 7.2. The machines were connected by a 100 Mb Ethernet switch. The *Discover Middleware* was installed on the machines running Windows NT 4.0, while the *Application* was installed on the machine running RedHat Linux 7.2. The *Discover portal* ran on the other machine running Windows NT 4.0. The following factors affect overhead of the DRBAC model.

- The number of roles assigned to the object.
- The frequency of the events (generated by the context agent at the object) that trigger transitions in the role state machine.
- The number of permissions assigned to each role.
- The frequency of the events (generated by the context agent at the subject) that trigger transitions in the permission state machine.

In the first set of experiments, we assigned each user 5 roles, and the role with highest privileges had 5 permissions. The events that triggered transitions in the role state machine were generated at different time interval. The times required to generate a request at the *Discover Portal* and get a response from the *Applications*, i.e. the interaction times, for different event frequencies are listed in Table 3. The first row is for the case without DRBAC.

In the second set of experiments, we randomly generate events to trigger transitions in the role state machine and vary the number of roles assigned. The role with the highest privileges is still assigned 5 permissions. Table 4 shows the interaction times for different number of roles.

In the last set of experiments, the user had a state machine with 5 roles and the role with the highest privileges was set as the active role. Events were randomly generated at the application server to trigger transitions in the permission state machine. The number of permissions assigned to the active role was varied. The interaction times for different number of permissions are listed in Table 5.

Table 3. Interaction time in ms. for different context event frequencies.

Event frequency	Time (ms.)
-	2300
1min	4732
2min	4403
3min	4102
4min	3482
5min	3104

Table 4. Interaction time in ms. for different number of roles.

Number of Roles	Time (ms.)
-	2300
5	2520
6	2608
7	2804
8	2920
9	3004

Table 5. Interaction time in ms. for different number of permissions.

Number of Permissions	Time (ms.)
-	2300
5	2500
6	2602
7	2698
8	2804
9	2912

These preliminary results show that in general the overheads of the DRBAC implementation are reasonable. The primary overheads were due to the event generated by the context agent - the higher the frequency, the larger was the overhead. The context agent can be implemented as an independent thread and as a result, the transition overheads at the object and subject are not significant.

5 Summary and Conclusions

In this paper, we presented the SESAME dynamic context-aware access control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underling dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC). A prototype implementation of SESAME and the DRBAC model within the Discover computational collaboratory was presented. The feasibility, performance and overheads of SESAME were experimentally evaluated. The results show that the overheads of the model are reasonable and the model can be effectively used for dynamic context-aware access control for Grid applications.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, 15(3): pp.200-222, 2001.

[2] R. Sandhu, E. Coyne, H. Feinstein and C. Youman, "Role-Based Access Control Models", IEEE Computer, 29(2): pp.38-47, 1996.

[3] L. Pearlman, V. Welch, I. Foster, C. Kesselman and S. Tuecke, "A Community Authorization Service for Group Collaboration", Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, p.0050, Monterey, CA, 2002.

[4] W. Johnston, S. Mudumbai, and M. Thompson. "Authorization and Attribute Certificates for Widely Distributed Access Control", Proceedings of IEEE 7th International Workshops on Enabling Technologies: Infrastructures for Collaborataive Enterprises, Stanford University, CA, USA, 1998.

[5] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids", Proc. 5th ACM Conference on Computer and Communications Security Conference, pp.83-92, San Francisco, CA, USA, 1998.

[6] M. Lorch, D. Kafura, "Supporting Secure Ad-hoc User Collaboration in Grid Environments", Proceedings of the 3rd Int. Workshop on Grid Computing, Springer Press, pp.181-193, Baltimore, MD, USA, November 2002

[7] L. Ramakrishnan, H. Rehn, J. Alameda, R. Ananthakrishnan, M. Govindaraju, A. Slominski,K. Connelly, V. Welch, D. Gannon, R. Bramley, and S.Hampton,"An Authorization Framework for a Grid Based Component Architecture",Proceedings of the 3rd International Workshop on Grid Computing, Springer Press, pp. 169-180, Baltimore, MD, USA, November 2002

[8] D. F. Ferraiolo , J. F. Barkley and D. R. Kuhn, "A Role Based Access Control Model and Reference Implementation Within a Corporate Intranet", ACM Transactions on Information and System Security, 2(1): pp.34-64, 1999.

[9] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control", ACM Transactions on Information and System Security, 4(3): pp.224-274, 2001.

[10] V. Bhat and M. Parashar, "A Middleware Substrate for Integrating Services on the Grid", Technical Report Number TR-268, Center for Advanced Information Processing, Rutgers University, November 2002.

[11] V. Mann, V. Matossian, R. Muralidhar and M. Parashar, "Discover: An Environment for Web-based Interaction and Steering of High-Performance Scientific Applications", Concurrency and Computation: Practice and Experience, John Wiley and Sons, 13(8-9) :pp.737-754, 2001.

[12] V. Mann and M. Parashar, "Engineering an Interoperable Computational Collaboratory on the Grid", Special Issue on Grid Computing Environments, Concurrency and Computation: Practice and Experience, John Wiley and Sons, 14(13-15): pp.1569-1593, 2002.

[13] R. Muralidhar and M. Parashar, "A Distributed Object Infrastructure for Interaction and Steering", In Concurrency and Computation: Practice and Experience, John Wiley and Sons, to appear.

[14] A. K. Dey, G. D. Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications", In Proceedings of Human Factors in Computing Systems: CHI 99, Pittsburgh, PA: ACM Press, pp.434-441, May 1999.

[15] Global Grid Form Web Site, http://www.ggf.org/, 2003.

[16] Globus Project Web Site, http://www.globus.org, 2003.

[17] Network Weather Service, University of California, Santa Barbara, Research Project Web Site, http://nws.cs.ucsb.edu/, 2003.