Role-Based Access Control for Cyber-Physical Systems Using Shibboleth

Vineela Muppavarapu and Soon M. Chung
Department of Computer Science and Engineering
Wright State University
Dayton, Ohio 45435, U.S.A

{muppavarapu.2, soon.chung}@wright.edu

Abstract

In this paper, we propose a role-based access control (RBAC) system for the distributed resources in a cyber-physical system. Current identity-based access control systems cause substantial administration overhead for the resource managers in the cyber-physical system because of the direct mapping between individual users and the access privileges on the resources. Our RBAC system uses Shibboleth, which is an attribute authorization service currently being used in Grids. The administration overhead is reduced in our system because the role privileges of individual users are managed by Shibboleth, not by the resource managers. For the uniform specification of access control policies across different security domains, we use the core and hierarchical RBAC profile of the eXtensible Access Control Markup Language (XACML); and for distributed administration of those policies, we used the Object, Metadata and Artifacts Registry (OMAR). OMAR is based on the e-business eXtensible Markup Language (ebXML) registry specifications developed to achieve interoperable registries and repositories. Our RBAC system provides scalable and fine-grain access control and allows privacy protection. Performance analysis shows that our system adds only a small overhead to the existing security infrastructures.

Keywords: Role-based Access Control (RBAC); Shibboleth; eXtensible Access Control Markup Language (XACML); Object, Metadata and Artifacts Registry (OMAR)

1. Introduction

Supporting role-based access control (RBAC) [1, 2] is desirable in cyber-physical systems, where the resources and their users may belong to different organizations. Both users and resources change regularly in these systems, and hence the system should support the dynamic nature of users and resources. Data security and privacy should be ensured as data is accessed across the institutional domains.

RBAC shows clear advantages over traditional discretionary and mandatory access control models in such environments, because it allows a uniform representation of diverse security policies [3]. In RBAC, permissions are associated with roles, and users are made members of appropriate roles, thereby acquiring the roles' permissions [4]. In a cyber-physical system with a large number of users, we could think of several groups of users, each with different level of access privileges via global roles across the whole system. When a global role of a user is mapped to a local role for a specific resource by the resource manger, it will specify the privileges the user can have; for example, access to a specific data file. RBAC can support a wide range of security policies using role privileges, role hierarchies, and constraints.

The typical identity-based authorization used today is not scalable because authorization information should be maintained for each user. In RBAC, authorization information is associated with roles, not with individual users. It has been shown that the cost of administering RBAC is proportional to $U+P$ per role, where U is the number of individuals in a role and P is the number of permissions required by the role, while the cost of associating users directly with permissions is proportional to $U \times P$ [5].

RBAC is distinguished by its inherent support for the Principle of Least Privilege [7], which requires that a user be given no more privileges than necessary to perform a task [1]. It can be easily enforced by first identifying the roles correctly and then assigning only those privileges to each role that allow the role members to perform their tasks. Users can request a particular role among those they are entitled to and hence gain the specific privileges tied with that role.

In certain instances, a user may wish to delegate only a subset of his/her rights to an application to act on his/her behalf. This requirement may arise in systems where a limited trust relationship is established between entities. For example, a user may contact a data mining service to mine certain data sets that the user has access to. If the trust between the user and the service is limited, then the user may want to delegate only a specific subset of his/her rights to the service, thus enabling it to complete only the required task and nothing more. With RBAC, such delegation could be done easily. For example, a user in a special role can delegate some of the privileges to other roles.

One of the key features of RBAC is the ability to manage itself through administrative roles and permissions [16]. Users in administrative roles can create roles and role hierarchies; make user-role and permission-role assignments; and specify constraints. Furthermore, they can grant administration privileges to other users.

In this paper, we propose a new RBAC system for heterogeneous resources in a cyber-physical system by using Shibboleth [8]. Shibboleth is an attribute authorization service and is designed to provide user attributes to the resources for access control, and it mainly targets the Internet-based resources. In our system, it is also used as a Role Enablement Authority (REA), which is responsible for assigning roles to users and for activating roles within the user's session [10].

Shibboleth allows pseudonymous interaction between users, thus protecting individual privacy [6]. The Shibboleth service maps a user name and attributes onto a unique identifying handle. To protect the user's privacy, the Shibboleth service can restrict the information about the holder of the handle, depending on who is asking for the information.

To specify the access control policies, we used the eXtensible Access Control Markup Language (XACML) [10]. XACML is a standard of the Organization for the Advancement of Structured Information Standards (OASIS) for describing the access control policies uniformly across different security domains [11]. Our system is based on the core and hierarchical components of the ANSI-RBAC [12], and the core and hierarchical RBAC profile of XACML defines how they can be specified in XACML.

For the distributed storage and administration of XACML policies and the user-role assignments, we used the OMAR. OMAR is an open-source implementation from the freebXML.org, and it provides an implementation of the OASIS e-business eXtensible Markup Language (ebXML) registry specifications [15]. A registry is an information system that securely manages any content type and the standardized metadata that describes the content. The ebXML registry specifications are developed to achieve interoperable registries and repositories with an interface that enables submission, query and retrieval [15].

OMAR supports the federation of registries, so that multiple registries can be linked together seamlessly and appear as a single logical registry, while retaining local autonomy and security. Thus, XACML-based policies can be stored and managed across multiple sites, while each site maintains its own registry and Shibboleth service. Hence, there is less chance to have a bottleneck, and there is no single point of failure in the system as authorization queries are distributed among the Shibboleth services.

2. Our RBAC System with Shibboleth

In our system, shown in Figure 1, Shibboleth maintains the security policies of the cyber-physical system, grants the users' memberships on the global roles, and then authorizes them to use the privileges of those roles. Thus, the assignment and activation of global roles can be handled by the user's home institution. The resource managers then only need to maintain the mapping information from the global roles to the local roles and local policies, thereby reducing the number of entries to be maintained in the role-map file dramatically. Our system also allows the specification of policies at the global level, thus if the users do not possess the required privileges, their access can be denied at the global level. This eliminates unnecessary authentication, mapping and connection overheads on the resource managers. When users join/leave the system, Shibboleth can just grant/revoke their memberships on the global roles.

The data flow in our system is described in detail as follows:

1. The client sends an attribute query to Shibboleth.
2. To obtain the user information, Shibboleth queries the OMAR registry [9], which stores the user-role assignments and the global access control policies in XACML.
3. The user attributes are returned from the OMAR registry.
4. Shibboleth issues user attributes in the form of Security Assertion Markup Language (SAML) [13] assertion. The SAML assertion is valid for a limited period of time specified by the *Conditions* statement in the assertion.

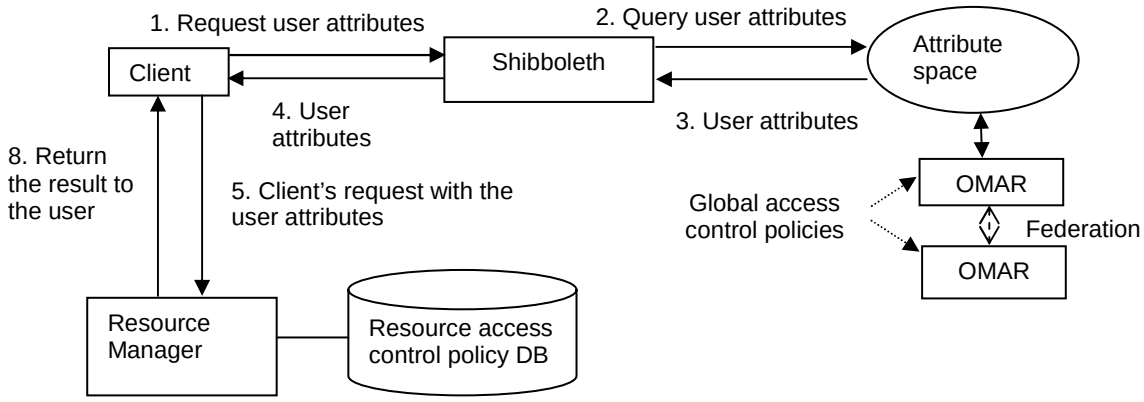

Figure1. Proposed RBAC system architecture

5. The client makes an access request to the resource manager along with the user attributes.
6. The resource manager obtains the user attributes from the client, and then queries the resource access control policy database to obtain the corresponding access control information, including the mapping of the global role to local roles and their access permissions.
7. If the access is allowed, then the client's request is performed by the resource manager, otherwise it is denied.

Our system works in the push model, as the user can directly obtain the permissions from the authorization service and then pass them to the resource manager at the time of making a request. The resource manager then verifies the authenticity of the user and authorizes the user based on the permissions obtained, provided the authority that issued them is trustworthy. An advantage of the push model is that it allows the user to explicitly select a role from the set of roles he/she is entitled to.

Additionally, the push model is inherently scalable because the resource manager does not need to contact the authorization service to obtain the user attributes for each access request. In case that the user and the authorization service belong to the same organization and are protected by a firewall, the push model should be deployed because the resource manager may not be able to contact the authorization service directly.

3. Performance Analysis

For the performance analysis, we implemented our proposed RBAC system on the Storage Resource Broker (SRB) [14], which a middleware for integrating heterogeneous data resources. The SRB stores the access control information of individual users in the Metadata Catalog (MCAT) database. The performance our RBAC system was analyzed in terms of the overheads incurred, compared with the existing implementation of SRB which does not use Shibboleth for access control. For our implementation, we installed an MCAT-enabled SRB server on a SuSE Linux machine with a 2.6 GHz Intel Pentium IV processor and 1 GB RAM.

SRB version 3.4.2 is used, and Oracle 9.0 is used to manage the MCAT database. The client is implemented on another SuSE Linux machine with a 1.6 GHz Intel dual-core processor and 2 GB RAM. Shibboleth identity provider (IdP) 1.3c was configured to run on the SSL-enabled Apache 2.2.0 and Tomcat 5.0.28 servers. We used OMAR 3.0 beta 1 as the repository for storing the XACML policies. Our client machine and the SRB server are connected by a Fast Ethernet LAN. For the purpose of analysis, the *Sls* command was invoked by the client, which is an SRB command line utility for listing the files/directories at a site in the SRB system.

The times taken to execute the *Sls* command with the original implementation of SRB and the modified implementation with Shibboleth are shown in Figure 2. The time difference between the original and modified implementations is about 20 milliseconds. So, we can claim that our RBAC system adds a very small overhead, compared to the total execution time required by the original implementation of SRB.

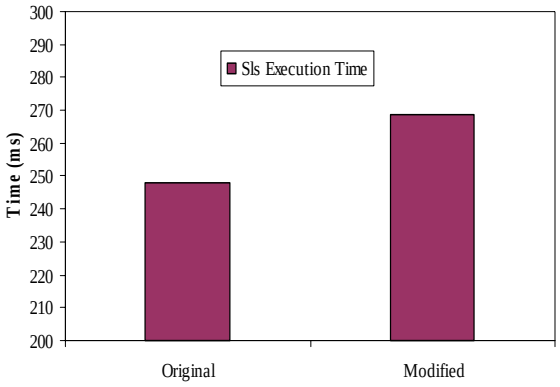

Figure 2. Total execution time for the *Sls* command

4. Conclusion

In this paper, we propose a role-based access control (RBAC) system for the distributed resources in a cyber-physical system. Our system supports privacy protection and reduces the administration overhead because the role privileges of individual users are managed by Shibboleth, not by the resource managers. XACML is used for uniform specification of access control policies across different security domains, and OMAR is used for distributed administration of those policies.

References

1. D. Ferraiolo and R. Kuhn, "Role-based Access Control," *Proc. of the 15th National Computer Security Conference*, 1992.
2. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.
3. J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, "Access-Control Language for Multidomain Environments," *IEEE Internet Computing*, vol. 8, no. 6, pp. 40–50, 2004.
4. C. Ramaswamy and R. S. Sandhu, "Role-based Access Control Features in Commercial Database Management Systems," *Proc. of the 21st National Information Systems Security Conference*, 1998.
5. D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn, "A Role-based Access Control Model and Reference Implementation Within a Corporate Intranet," *ACM Trans. on Information and System Security*, vol. 2, no. 1, pp. 34–64, 1999.
6. V. Welch, T. Barton, K. Keahey, and F. Siebenlist, "Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration," *Proc. of the 4th Annual PKI R&D Workshop*, 2005.
7. T. Mayfield, J. E. Roskos, S. R. Welke, and J. M. Boone, "Integrity in Automated Information Systems," Technical Report, National Computer Security Center, 1991.
8. S. Carmody, "Shibboleth Overview and Requirements," Shibboleth Working Group Document, http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html, 2001.
9. Object, Metadata and Artifacts Registry, http://ebxmlrr.sourceforge.net/3.0/
10. *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*, Organization for the Advancement of Structured Information Standards (OASIS), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf, 2005.
11. *eXtensible Access Control Markup Language (XACML) Version 2.0*, Organization for the Advancement of Structured Information Standards (OASIS), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, 2005.
12. Secretariat of Information Technology Industry Council (ITI), "American National Standard for Information Technology – Role Based Access Control," http://csrc.nist.gov/rbac/rbac-std-ncits.pdf, 2003.
13. *Assertions and protocols for the OASIS Security Assertion Markup Language (SAML) v1.1*, Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, 2003.
14. C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker," *Proc. of Conference of the Centre for Advanced Studies on Collaborative Research*, 1998.
15. ebXML registry technical committee, Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep.
16. R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 Model for Role-based Administration of Roles," *ACM Trans. on Information and System Security*, vol. 2, no. 1, pp. 105–135, 1999.