Role-Based Authorization in Decentralized Health Care Environments

Gail-Joon Ahn
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC 28223, USA
gahn@uncc.edu

Badrinath Mohan
Laboratory of Information Integration, Security and Privacy (LIISP)
University of North Carolina at Charlotte
Charlotte, NC 28223, USA
bmohan@uncc.edu

ABSTRACT

The formation of a distributed system is based on a collection of distributed components and it requires the ability for components to exchange syntactically well-formed messages. To simplify network programming for such interactions and to realize security services for those components, we need a component-based software architecture that enables software components to communicate directly over a network in a reliable and efficient manner. One of those models is Distributed Component Object Model (DCOM) which is used for interacting with distributed components within the local intranet. In this paper, we overview an aspect of DCOM concerning software architecture and access control. And we describe the concept of role-based access control (RBAC) which began with multi-user and multi-application on-line systems pioneered in the 1970s. Also we investigate how we can enforce the role-based access control as a security provider within the critical environment such as health care industry accessing distributed components legitimately. We demonstrate the feasibility of our approach through a proof-of-concept prototype implementation.

Keywords

Authorization, Role-based, Health care envrionments

1. INTRODUCTION

New design and deployment issues have been introduced by distributed applications. And these applications demand a robust infrastructure for distributed computing. Nowadays many applications are distributed, in the sense that they have at least two components running on different machines. But because these applications were not designed to be distributed, they are limited in scalability and ease of deployment. Any kind of workflow or groupware application, most client/server applications, and even some desktop productivity applications essentially control the way their users communicate and cooperate. Adopting the notion of distribution into these applications benefits the user and optimizes the use of network and computer resources. The application designed with distribution in mind can accommodate different clients with different capabilities by running components on the client side when possible and running them on the server side when necessary.

The proposed rules of the Health Insurance Portability and Accountability Act (HIPAA), circulated by the U.S. Department of Health and Human Services (HHS) through the Health Care Financial Administration (HCFA) strongly require the services of security and privacy. Along with this movement, a secure solution for the complex environment like health care industry has been extremely demanded. However, it is not easy to come up with a standard secure information system for the highly decentralized health care industry.

The Robert Wood Johnson Foundation has provided a grant to a five-state coalition (including North Carolina) to develop the HealthKey projects that demonstrate how security can be complied with clinical and health care information systems. Most projects have been focusing on authentication mechanisms. As a result, a scalable authentication mechanism has been developed and deployed to modernize health care systems [3]. In addition, we identified the need of access control for the health care environment [3]. One of challenges in this project was to demonstrate how access control can be achieved for the health care environment. We first investigated the information technology (IT) infrastructure of the health care environment. According to our study, most health care applications are distributed applications and run on Microsoft's Windows platforms. And there is a requirement such that the deployment of access control solution is to have minimal changes to the existing IT infrastructure. To develop an access control mechanism for these distributed applications, we may consider several component-based architectures and frameworks such as DCOM (Distributed Component Object Model), CORBA (Common Object Request Broker Architecture), and .NET. Among these frameworks, DCOM is a solution to support their IT infrastructure fulfilling their requirement while .NET and CORBA approaches require upgrades and modifications to their existing IT infrastructure, respectively. Various forms of access controls (e.g., identiy-based, label-based and role-based) have been evaluated for

SAC 2003, Melbourne, Florida, USA

use in health care. In this work, we adopt role-based access control which has been widely accepted to accommodate organizational roles.

In this paper, we demonstrate how we can enforce role-based access control in DCOM with advantages that RBAC allows us to achieve, we use DCOM's group mechanism which is similar to the notion of roles in RBAC. Even though DCOM does not have role concept inside, it is possible for RBAC to be applied to DCOM security mechanism, which can satisfy DCOM's access security policy and achieve ease of administration.

This paper begins with the description of the DCOM model in section 2. In section 3, we discuss the RBAC model following by section 4 which includes conceptual integration between RBAC model and DCOM. Section 5 describes our implementation in detail and Section 6 concludes the paper.

2. DISTRIBUTED COMPONENT OBJECT MODEL

In programming and engineering disciplines, a component is an identifiable part of a larger program or construction. Usually, a component provides a particular function or group of related functions. In programming design, a system is divided into components that in turn are made up of modules. In object-oriented programming and distributed object technology, a component is a reusable program building block that can be combined with other components in the same or other computers in a distributed network to form an application. Examples of a component include: a single button in a graphical user interface, a small interest calculator, an interface to a database manager. Components can be deployed on different servers in a network and communicate with each other for needed services [10].

COM is Microsoft's framework for developing and supporting program component objects. The COM provides a set of interfaces allowing clients and servers to communicate within the same computer (running a Windows 95 or NT system). It is aimed at providing similar capabilities to those defined in CORBA, the framework for the interoperation of distributed objects in a network. Whereas OLE provides services for the compound document that users see on their display, COM provides the underlying services of interface negotiation, life cycle management, licensing, and event services. DCOM is a protocol that enables software components to communicate directly over a network in a reliable, and efficient manner [12]. At the same time it is a program interface in which client program objects can request services from server program objects on other computers in a network. For example, one can create a page for a Web site that contains a script or program that can be processed (before being sent to a requesting user) not on the Web site server but on another, more specialized server in the network. Using DCOM interfaces, the Web server site program (now acting as a client object) can forward a Remote Procedure Call (RPC) to the specialized server object, which provides the necessary processing and returns the result to the Web server site. This result is passed on to the Web page viewer.

A client that needs to communicate with a component reside on different machines cannot call the component directly, but has to use some form of network communication provided by the operating system. DCOM provides

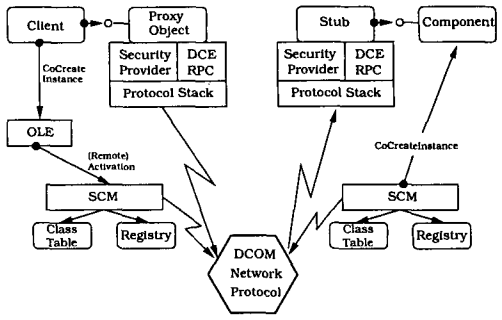

Figure 1: DCOM Architecture

this communication in a completely transparent fashion: it intercepts calls from the client and forwards them to the component with a network protocol. Neither the client nor the component are aware that the wire that connects them has just become a little longer. Figure 1 shows the overall DCOM architecture: the COM run-time provides object-oriented services to clients and components and uses RPC and the security provider to generate standard network packets that conform to the DCOM.

Different platforms use different security providers, and many platforms even support multiple security providers for different usage scenarios or for interoperability with other platforms [4, 11]. DCOM and RPC are designed in such a way that they can simultaneously accommodate multiple security providers. All these security providers provide a means of identifying a security principal, a means of authenticating a security principal, and a central authority that manages security principals and their keys. DCOM uses the extensible security framework provided by Windows NT. Widows NT provides a solid set of built-in security providers that support multiple identification and authentication mechanisms, from traditional trusted-domain security models [5] to noncentrally managed, massively scaling public-key security mechanisms [1]. A central part of the security framework is a user directory, which stores the necessary information to validate a user's credentials. DCOM can make distributed applications secure without any security-specific coding or design in either the client or the component. Just as the DCOM programming model hides a component's location, it also hides the security requirements of a component.

The most obvious security requirement on distributed applications is the need to protect objects against unauthorized access. Sometimes only authorized users are supposed to be able to connect to an object. In other cases, non-authenticated or unauthorized users might be allowed to connect to an object, but must be limited to certain areas of functionality. Current implementations of DCOM provide declarative access control on a per-process level. Another related requirement on a distributed infrastructure is to maintain control over who can create objects. Since all COM objects of a machine are potentially accessible via DCOM, it is critical to prevent unauthorized users from creating instances of these objects. The scope of this paper will be within this issue, *access security.*

3. OVERVIEW OF RBAC MODEL

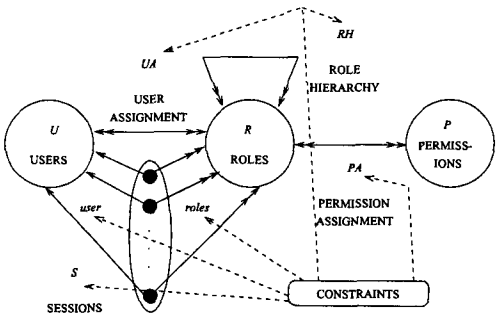

Figure 2: RBAC Model

RBAC is an alternative policy to traditional mandatory access control (MAC) and discretionary access control (DAC) [18]. As MAC is used in the classical defense arena, the policy of access is based on the classification of objects such as top-secret level [16]. The main idea of DAC is that the owner of an object has discretionary authority over who else can access that object [8]. But RBAC policy is based on the role of the subjects and can specify security policy in a way that maps to an organization's structure.

A general family of RBAC models called RBAC96 was defined by Sandhu et al [17]. Figure 2 illustrates the most general model in this family. Motivation and discussion about various design decisions made in developing this family of models is given in [17, 13]. Also, there are variations regarding distributed systems [19].

Figure 2 shows (regular) roles and permissions that regulate access to data and resources. Intuitively, a user is a human being or an autonomous agent, a role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role, and a permission is an approval of a particular mode of access to one or more objects in the system or some privilege to carry out specified actions. Roles are organized in a partial order $\geq$, so that if $x \geq y$ then role x inherits the permissions of role y. Members of x are also implicitly members of y. In such cases, we say x is senior to y. Each session relates one user to possibly many roles. The idea is that a user establishes a session and activates some subset of roles that he or she is a member of (directly or indirectly by means of the role hierarchy). The RBAC model has the following components and these components are formalized from the above discussions.

- U is a set of users,
- R is disjoint sets of roles and administrative roles respectively,
- P is disjoint sets of permissions and administrative permissions,
- $UA \subseteq U \times R$, is a many-to-many user to role assignment relation,
- $PA \subseteq P \times R$ is many-to-many permission to role assignment relations,
- $RH \subseteq R \times R$ is partially ordered role hierarchies (written as $\geq$ in infix notation),
- S is a set of sessions,
- $user : S \rightarrow U$, is a function mapping each session s_i to the single user $user(s_i)$ and is constant for the session's lifetime,
- $roles : S \rightarrow 2^R$ is a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$ (which can change with time) so that session s_i has the permissions $\cup_{r \in roles(s_i)}\{p \mid (\exists r'' \leq r)[(p, r'') \in PA]\}$, and
- there is a collection of constraints stipulating which values of various components of the RBAC model are allowed or forbidden.

A user can be a member of many roles and a role can have many users. Similarly, a role can have many permissions and the same permissions can be assigned to many roles. Each session relates one user to possibly many roles. Intuitively, a user establishes a session during which the user activates some subset of roles that he or she is a member of the permissions available to the users are the union of permissions from all roles activates in that session. Each session is associated with a single user. This association remains constant for the life of a session. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance. Each session may have a different combination of active roles. The concept of a session equates to the traditional notation of a subject in access control. A subject is a unit of access control, and a user may have multiple subjects (or sessions) with different permissions active at the same time.

4. AUTHORIZATION WITHIN THE HEALTH CARE ENVIRONMENT

Today's health care industry is striving to achieve the development and deployment of computer-based patient records for improvements in health care quality, cost, and access, Towards this objective, vendors are developing and health care organizations are selecting and implementing computer applications and comprehensive systems which will support these efforts. These systems are being evaluated based on a number of essential criteria; one of which is their ability to support the security requirements necessary to ensure the integrity and confidentiality of health information to protect the privacy and certain legal records of patients, providers, and health care organizations. The Institute of Medicine defines a computer based patient record as "an electronic patient record that resides in a system designed to support users through availability of complete and accurate data, practitioner reminders and alerts, clinical decision support systems, links to bodies of medical knowledge, and other aids." The Institute of Medicine also defined the computer-based patient record system as "the set of components that form the mechanism by which patient records are created, used, stored, and retrieved ... [including] people, data, rules and procedures, processing and storage devices, and communication and support facilities." [6]

A computer-based patient record is electronic maintained information about an individual's lifetime health status and health care. It replaces the paper medical record as the primary record of care, meeting all clinical, legal, and administrative requirements. It is more than the content of

today's paper medical record. Information technology can merge text with images and sound, It facilitates access, upon patient authorization, for legitimate users, to health data stored in multiple, dispersed locations. Such access to data contributes to continuity of health care. In today's dynamic health care environment, disparate information systems are coming together to form computer-based patient record systems intent on supporting the ongoing maintenance of a longitudinal record for quality patient care. Computer-based patient record systems must include a variety of security features to protect the confidentiality and integrity of health information to preserve the privacy and property rights of patients, providers (individual caregivers and enterprises), and others who interact with the computer-based patient record.

Complete and accurate data, reduction in redundant data collection, and elimination of repetitive testing contribute to productivity gains, reduced rest, and improved management of the health care delivery system. The essential features of a secure system and network may be categorized as: authentication, authorization, audits trails, and secure data storage and transmission.

Although the importance of authorization in health care information systems has been recognized, they have not received much attention as a security feature, while other issues such as authentication, audit, and secure transmission have been practiced and discussed at considerable length. This observation motivates this work such that a systematic and practical form of access control should be developed and evaluated for use in health care.

5. ROLE-BASED AUTHORIZATION MANAGEMENT USING DCOM

In this section we outline one approach to enforcing RBAC in DCOM. For some applications, a single component-wide access control list is not sufficient. Some methods in a component may be accessible only to certain users. For example, an accounting business component may have a method for registering new transactions and another method for retrieving existing transactions. Only members (such as Bob) of the accounting department (user group "Accounting") should be able to add new transactions, while only members (such as Alice) of transaction management (user group "Transaction") should be able to view the transactions. And members of top management (user group "Top Management") are able to add and view the transaction. How can an application use DCOM security to implement the selective security required in this example?

We can approach this example with programmatic control using DCOM [7, 9]. the client is impersonated by DCOM. After this, the called thread can perform only those operations on secured objects, that the client is permitted to perform. The component can then try to access a secured object, such as a registry key, that has an Access Control List on it. If this access fails, the client was not contained in the ACL, and the component rejects the method call. By choosing different registry keys according to the method that is being called, the component can provide selective security in a simple way.

We can simulate RBAC in DCOM along the lines of this example[1]. Firstly we try to simulate this with a conceptual

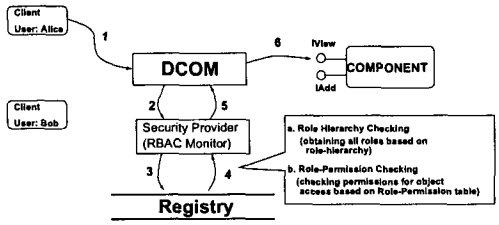

Figure 3: Integration with RBAC

abstraction such that RBAC can be inserted into DCOM architecture (in Figure 1) as part of security provider. In order to use RBAC96, we should accommodate role-hierarchies. Roles would map to NT user group(s) (which do not support hierarchies). For example, user group "Accounting" would map to role "*Accounting*". We can represent the role-relationship as role-hierarchy. The *Accounting* role can have permission to add new transactions, while only the *Transaction* role can have permission to view the transactions. The *Top Management* role is senior to *Accounting* and *Transaction* and thereby inherits all permissions from junior roles.

In Figure 3 we overview a unified checking mechanism using RBAC proposed by [2]. For ease of reference, we call the portion of security provider that enforces role-based access control as RBAC monitor. At the role hierarchy checking step, the client's all roles that he or she is a member of (by means of role hierarchies) should be checked. After this step RBAC monitor decides whether the client can have permission(s) to access the component according to given role(s) during the role-permission checking. For example, assume that a client Chris has a role *Top Management* and tries to access an accounting business component. During role hierarchy check, we can know Chris's role membership: {*Accounting*, *Transaction*, *Top Management*}. After the role permission check, he can have all permissions from junior roles such as *Accounting* and *Transaction* using Registry Key. This means Chris can add new transactions and also view the transaction. Let's consider that Bob has a role *Accounting* and tries to access an accounting business component. We can simply know that Bob can only add new transactions but he can not view the transaction. The RBAC approach can also reduce the interactions between a component and secure object such as Registry.

5.1 DCOM

This section briefly describes DCOM procedures and configuration with sample codes of our implementation that has been programmed by Visual C++/MFC. The client calls CoInitialize which initializes the COM library for use. Next, the client calls COM library's CoCreateInstance, passing the ID for the desired class (CLSID) and an array of desired interface IDs (IIDs). The call CoInitialize uploads the COM library entirely:

```
HRESULT hr = CoInitialize (NULL);
```

Using the server machine address, the client SCM establishes an RPC link with the server machine's SCM, passing it the desired CLSID and IID(s). The server SCM checks its

[1]We assume that role engineering, such as user-role and permission-role assignments, is done by [14, 15], including the simulation of role-hierarchies.

Class Table to see if the server is running, and if it isn't, it looks up the CLSID in the Registry to find out what command needs to be executed to start the server, and executes it. When a COM server starts up, the first thing it calls is CoInitialize. The second thing it calls is CoRegistryClassObject, passing the implemented CLSID and a pointer to the server program's Class Factory. This effectively advertises the server program to DCOM, adding its entry into the Class Table. The server machine can be specified as below:

```
// Remote server info
COSERVERINFO cs;
memset(&cs, 0, sizeof(cs));
// Allocate the server name
// in the COSERVERINFO struct
cs.pwszName = _bstr_t(name); ... ...
typedef struct  _COSERVERINFO
    {
    DWORD dwReserved1;
    LPWSTR pwszName;
    // Pointer to the name of
    // the machine to be used.
    COAUTHINFO  *pAuthInfo;
    // To override the default
    // activation security.
    DWORD dwReserved2;
    }   COSERVERINFO;
```

Whenever a client requests a new instance of a server class, the SCM calls a method called CreateInstance on the server program's Class Factory. This is the common gateway used by all clients. CreateInstance then instantiates the object and passes the pointer to the object's IUnknown interface back to the SCM. Once the SCM has a pointer to the object's IUnknown interface, it can use QueryInterface to request pointers to other interfaces. This multiple query interface (MQI) can be realized in the codes as follows:

```
// structure for CoCreateInstanceEx
MULTI_QI qi[2];
memset(qi, 0, sizeof(qi));
// Fill the qi with a valid interface
qi[0].pIID = &IID_ICheck;
qi[1].pIID = &IID_ISomeOtherInterface;
// get the interface pointer
hr = CoCreateInstanceEx(
    CLSID_Check,  //clsid
    NULL,  //outer unknown
    CLSCTX_SERVER,  //server context
    &cs,  //server info
    2,  //size of qi
    qi); //MULTI_QI array
... ...
typedef struct tagMULTI_QI {
    const IID *pIID;
    IUnknown *pItf;
    HRESULT hr; } MULTI_QI;
```

Data passed back between client and server has to be packaged into RPC packets. This process is called marshalling, and is performed by the Proxy/Stub DLL(s). Using the Interface ID (IID) returned from the MQI step, the SCMs on each machine interrogate the Registry for the DLL(s) that need to be loaded into the client's and server's process spaces. Interface Pointers are then handed back to the client

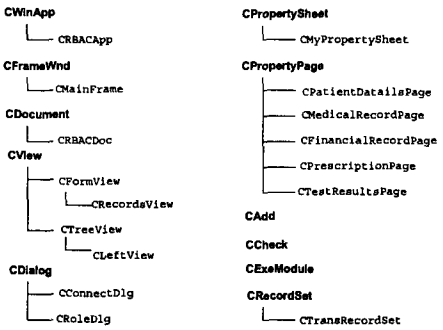

Figure 4: Classes Diagram

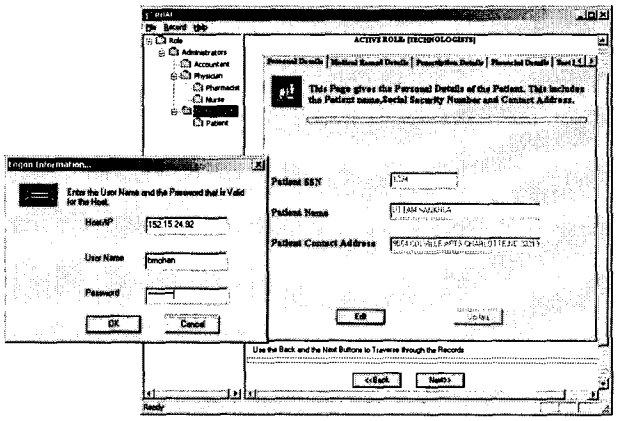

Figure 5: A Role-based Heath Care Information System

that issued the CoCreateInstance call. Using the Interface Pointer, the client finally can access a remote method on the server.

5.2 Role-based Authorization

In order to enforce our framework described in this paper, we also need to construct role-permission relation. Let's consider the health care information system which has role-permission table illustrated in table 1, and deals with critical information such as medical records. Prescription, Test-Result, Medical Record, and Financial Record are objects which include the information accessed by users who have permissions to access those objects. And each permission is an interface for each object. Table 1 implies that *Physician* role can have `read` and `write` permissions to objects such as Prescription and Medical Record, and he/she can just read the object Test Result. And *Patient* role can just read only his/her objects such as Prescription, Medical Record, Test Result, and Financial Record. That is, this table displays the relation between roles and permissions. Even though this table is not the exhaustive example, it is sufficient to show the need of role-permission table. This kind of access policy allows us to prevent an unauthorized user from doing malicious thing on (with) the critical information.

Integrating with RBAC, we developed several Classes and Objects for our health care information system as shown in Figure 4. Based on DCOM configuration, RBAC information infrastructure and these classes, we developed a proof-

	Prescription	Test-Result	Medical Record	Financial Record
Physician	R W	R	R W	-
Nurse	-	-	R W	-
Pharmacist	R W	-	-	-
Technologist	-	R W	-	-
Accountant	-	-	-	R W
Patient	R	R	R	R

Table 1: An Example of Role-Permission Table

of-concept implementation to show the feasibility of our approach. Figure 5 illustrates a role-based health care information system.

6. CONCLUSION

In this paper, we have described that the adoption of Role-Based Access Control as part of security provider can simplify the access control and provide administrative convenience to highly decentralized health care environments. To support our approach we discussed the architecture and security mechanism of DCOM. Also, we briefly looked at the RBAC model as a security specification model. Finally, we have shown how RBAC can be accommodated in DCOM to provide access control for the health care environment. Even though our solution was to target distributed components within the local intranet, we believe that our approach can be extended for web-based health care applications over the Internet. We have been currently considering .NET technology for those applications because .NET is used for interaction of distributed components in the Internet as an extension of DCOM.

7. ACKNOWLEDGEMENTS

This work was partially supported by the Robert Wood Johnson Foundation and this work was also supported, in part, by funds provided by the University of North Carolina at Charlotte.

8. REFERENCES

[1] C. Adams and S. Lloyd. *Understanding Public-key Infrastructure: Concepts, Standards, and Deployment Considerations.* Macmillan Technical Publishing, November 1999.

[2] G.-J. Ahn. Role-based access control in DCOM. *Journal of Systems Architecture*, 46(13), November 2000.

[3] G.-J. Ahn. Scalable authentication architecture for critical information system. In *Proceedings of the 18th AFCEA (Armed Forces Communications and Electronics Association) Annual Federal Database Colloquium and Exposition*, San Diego, CA, August 28-30 2001.

[4] P. E. Ammann and J. C. Knight. Data diversity: An approach to software fault-tolerance. *IEEE Transactions on Computers*, 37(4):418–425, April 1988.

[5] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria*, December 1985. DoD 5200.28-STD.

[6] R. Dick and E. Steen. *The Computer-based Patient Record: An Essential Technology for Health Care.* National Academy Press, Washington, DC, 1991.

[7] R. Grimes. *Professional DCOM Programming.* WROX Press Ltd., 1997.

[8] S. Jajodia, P. Samarati, V. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings of the ACM SIGMOD international conference on management of data*, pages 474–485, 1997.

[9] Microsoft. *DCOM Architecture.* Microsoft Professional Developers Conference, 1997.

[10] M. Moriconi and et al. Secure software architectures. In *Proceedings of IEEE Symposium on Research in Security and Privacy.* IEEE, 1997.

[11] B. Randell. System structure for software fault tolerance. *IEEE Transactions on Software Engineering*, SE-1(12):220–232, June 1995.

[12] D. Rogerson. *Inside COM.* Micosoft Press, 1996.

[13] R. Sandhu. Rationale for the RBAC96 family of access control models. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control.* ACM, 1997.

[14] R. Sandhu and G.-J. Ahn. Decentralized group hieraches in UNIX: An experiment and lessons learned. In *Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, Arlington, VA, October 5-8 1998.

[15] R. Sandhu and G.-J. Ahn. Group hierarchies with decentralized user assignment in Windows NT. In *Proc. International Association of Science and Technology for Development (IASTED) Conference on Software Engineering*, Las Vegas, Nevada, October 1998.

[16] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[17] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[18] R. S. Sandhu and P. Samarati. Authentication, access control and intrusion detection. In A. B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 1929–1948. CRC Press, 1997.

[19] N. Yialelis, E. Lupu, and M. Sloman. Role-based security for distributed object systems. In *Proceedings of the IEEE Fifth Workshops on Enabling Technology: Infrastructure for collaborative enterprise.* IEEE, 1996.