Role Delegation in Role-Based Access Control

SangYeob Na
Computer Dept, Namseoul Univ
Maeju-ri 21, Sunghwan-eup
Chunan, Chung-Nam, KOREA
+ 82 0417 580 2106
nsy@nsu.ac.kr

SuhHyun Cheon
Computer Engineering Dept, Dongguk Univ
Phil-dong 7-26, Chung-gu
Seoul, KOREA
+82 02 2260 3343
shcheon@cakra.dongguk.ac.kr

ABSTRACT

In distributed-computing environments, applications or users have to share resources and communicate with each other to perform their jobs more efficiently. For better performance, it is important to keep resources and the information integrity from the unexpected use by unauthorized user. Therefore, there is a strong demand for the authentication and the access control of distributed-shared resources. Nowadays, three kinds of access control, discretionary access control (DAC) mandatory access control (MAC) and role-based access control (RBAC) have been proposed. In RBAC, there are role hierarchies in which a senior role can perform the permission of a junior role. However, it is sometimes necessary for a junior role to perform a senior role's permission, which is not allowed basically by a junior role. In this paper, we will propose a role delegation method, consisting of a role delegation server, and a role delegation protocols. We divide the delegation into two by the triggered object: active delegation and passive delegation. Consequently, a junior role can gain a senior role's permissions.

Keywords

Role-Based Access Control, Role Delegation, Delegation Server, Delegation Protocol, Active Delegation, Passive Delegation

1. INTRODUCTION

In the distributed client-server computing environment, users sharing resources and communicating with others can work more efficiently. With the increase of the shared information and resources in the distributed system, unauthorized access to the information by illegal users that leads to the leakage of the data also increases. To protect the information in a distributed computing environment, it is necessary to secure the data through the user authentication and access control policy. This kind of policy has to be offered transparently to users or application programs for the convenient use of the system.

Access control is classified into three kinds [14] - discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC).

MAC enforces access controls on the basis of information security labels attached to users and objects. It shows access control relationship that cannot be changed by the object's owner. MAC can determine all kinds of access controls between subjects and objects consistently. If some object is duplicated, access control relationship to the original object must be equally applicable to the duplicated object. Also the object's owner cannot change access control relationship. Security labels have to be granted to all subjects and objects by the system supervisor, and it can be changed only in accordance with the contents of the object.

In case of DAC, access control restricts the access to the object on the basis of the identity of the user or the group. The owner of the object determines access control relationship. Therefore, it is difficult to maintain access control consistency. Access control can be established in one subject-object unit, and users who have permissions can allow any other users to access to data. But, because access control policy can be changed at the owner's own discretion, and owners can optionally delegate their authorities to other subjects, it is difficult to control information efficiently. Information related to the meaning of data cannot be involved in DAC, because access control is only based on qualifications of subjects [2].

Compared with MAC and DAC, RBAC determines access permissions that roles can perform, and assigns roles to users. Users can access objects according to assigned roles to them. As a result, the organization can not only preserve access control policy appropriate to its characteristics consistently, but also maintain access control relationship between subjects and objects independently. Even if access control policy is changed, the new access rights have to be allowed not to the user but to the role itself. RBAC can manage the complicated security policy efficiently [6][7][8].

A role in RBAC is the aggregate of responsibility and authority, to which the access to the object is permitted [1]. Each role having relations with other roles exists in role hierarchies according to the access control policy. Senior roles inherit authorities of junior role [7][8]. In this case, the outstanding problem is that a junior role or the role, which is not included in hierarchies, cannot perform permissions of a senior role.

In this paper, a role delegation method will be proposed in which junior roles can be temporarily granted senior roles' permissions. Delegation server and delegation protocol also will be presented. The former decides whether it is possible to delegate roles or not, and the latter describes how delegation is performed. The delegation method is divided into active delegation and passive delegation. A medical institution is taken as an example, because recently there has been a lot of research to present the standard model of RBAC in a medical institution [9].

The rest of paper is organized as follows. Section 2 begins with descriptions of RBAC model and its characteristics. Section 3 presents the necessity for a role delegation, a delegation server and a delegation protocol. Differences between active delegation and passive delegation are also explained with examples. The final section gives our conclusions and future research.

2. Role-Based Access Control Model

Access control is what allows the certified user to access the inner information of a system within the limits of permission. Computer based access controls can prescribe not only who or what process may have access to a specific system resource, but also the type of access that is permitted [16].

Role-based access control (RBAC) is proposed and studied as an alternative for MAC and DAC [13]. In RBAC, it is possible to simplify the complicated form of an organization’s access control policy. Access decisions are based on the roles, which is part of an organization. RBAC is a non-discretionary access control in which the system administrator allows the role’s permissions to the user by defining user, role, and permission. The system administrator divides roles according to operations in an organization, and gives access permissions to roles. The administrator of the system or organization gives access permissions to roles, and users are endowed with roles according to their responsibility and obligation [4]. Users, who are granted a role in system, can manage their works with their role permissions. In case of changing access control policy, the system supervisor easily can grant a new permission or can eliminate the existing permission to the role. Because access permissions are granted to roles (permissions are associated with roles), not to users, it is possible to manage access control policy more efficiently. There are many variations of RBAC, but the basic architecture of RBAC is that permissions are assigned to roles (not directly to users) and roles are assigned to users [7][12].

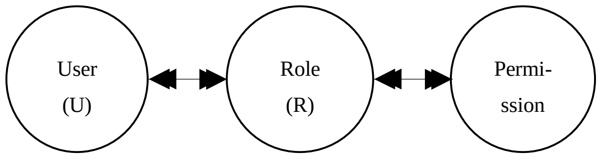
2.1 Role-Based Access Control Components

The basic components of RBAC model are User, Role and Permission [6][7][12]. User is a person, who uses the system or an application program within the system. Membership to roles is granted to users based on their obligation and responsibility in the organization. The operation of a user can be carried out based on the user’s role.

Role means a set of functional responsibilities within the organization. The system administrator defines roles, a combination of obligation and authority in organization, and assigns them to users. User-Role relationship represents collection of the user and role.

Permission is the way for the role to access to more than one resource, and is subdivided into obligation and authorization [1]. The former specifies which activities a subject is permitted (or forbidden) to perform on a set of objects, the latter represents which activities a subject must or must not perform on a target object. A subject means the user or the process that attempts to access resources, and a object represents resources.

Basic RBAC model consists of User-Role (U-R) relationship, Role-Permission (R-P) relationship and Role-Role (R-R) relationship. User-Role (U-R) relationship represents which user is assigned to perform what kind of role in the organization. The administrator decides the U-R relationship. When the user logins, the system U-R relationship is referenced to decide which role is executed.



[Figure 1] Basic Model of RBAC

In this article, Role-Role relationship and Role-Permission relationship will be explained for a role delegation method. In our paper, we simplify RBAC structure [7] in order to explain a role delegation.

2.2 Role-Permission(R-P) Relationship

Permission consists of obligation and authorization [1]. The former is the aggregate of operations that the pertinent role must perform or not, and the latter is the collection of operations that are permitted to roles or not. Expressions of obligation and authorization are simplified as defined in [4][17].

{identifier, mode, role, {action}, target, constraints, exception}

[identifier] : Uniquely identifying the permission

[mode] : o:obligation, a:authorization +:positive,-:negative

[role] : role which can process this permission

[action] : actual operation by permission

[target] : object which operation being applied

[constraints] : limit the applicability of the permission

[exception] : exceptional condition

In R-P Relationship, permission is applied Role Group (explained in section 2.3). Identifier is used to identify the permission. Mode specifies obligation and authorization, with either + and - denoting a positive or a negative policy respectively. e.g. o+ denotes a positive obligation. Role is to carry out this permission. Action means operation, which is to be processed by role, target is object influenced by action in permission. Constraints represent limits the applicability of the permission, e.g. a time period, or execution condition. Exception represents exceptional condition for permissions to be executed or not, e.g. if a permission has a-mode then this permission is not executed by the indicated role, but some specified exceptional condition occurred then permission can be allowed to be executed by the role.

For instance, obligation such as “a nurse must check the patient’s condition every morning at 8 o’clock” can be expressed as **{np1, o+, nurse, {Check}, patient, every 08:00, -}**. And authorization such as “a specialist can read chart of patient by intern” can be changed as **{dp1, a+, specialist, {read}, chart by intern, -, -}**. The example of Role-Permission relationship used in this paper is presented in [Table 1].

[Table1] Simplified Role-Permission Relationship Model

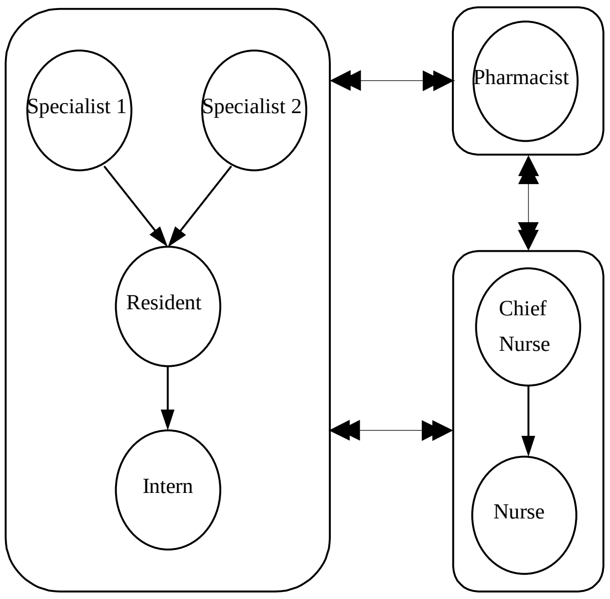
Role Group	Permission
Doctor	**{dp1, a+, specialist,{read,fix},chart by intern, -, -}**
	{dp2, o+, specialist,{chief of surgical operation}, patient, -,-}
	{dp3, a+, resident,{support of surgical operation}, specialist, request of specialist,-}
	{dp4, a-, resident,{read,fix}, chart by intern, -, no specialist}
	{dp5, a+, intern,{make}, chart for patient, -, -}
	{dp6, o-, intern,{chief of sugical operation}, patient, -, -}
Nurse	**{np1, o+, chief nurse,{assign}, nurse-patient,every 09:00, -}**
	{np2, a+, nurse,{injection by chart}, patient, by chart,-}
	{np3, a-, nurse,{preparation of medicine},drug,-,emergency}
Phar-macist	**{pmp1, a+, pharmacist, {preparation of medicine}, patient by chart, doctor request, -}**
	{pmp2, o+, pharmacist, {make report}, used drug, every 18:00, -}

2.3 Role-Role(R-R) Relationship

In the RBAC model, roles are organized in a hierarchical structure according to their permissions within the organization [5]. Roles with the similar authorization are managed as a group. The supervisor of organization classifies a lot of groups in order to manage roles. A role is subdivided into senior role and junior role within the hierarchical structure of role groups. There exists an inheritance relationship where a senior role may inherit the permissions of a junior role. It is possible to have multiple inheritance.

In [Figure 2], there are three role groups - a doctor group, a nurse group and a pharmacist group. The doctor group has three kinds of roles - specialist, resident and intern. Within the doctor group of the medical organization, residents and specialists (the senior roles) inherit permissions from interns. This means the each member of the role specialist naturally “contains” the permissions of residents and interns. The senior roles are at the top of the hierarchy with junior roles being at the bottom [8]. Senior role contains more permissions than junior role. The nurse group has two roles - chief nurse and nurse. And pharmacist role group has only one role.

In [Figure2], specialist is a senior role in doctor role group. Thus, specialist has the permission, which is granted to resident and intern, implicitly. In [Table 1], doctor group has six permissions explicitly. Specialist, resident and intern have two permissions for each explicitly. But, specialist has six permissions, resident has four permissions, and intern has two permissions implicitly.



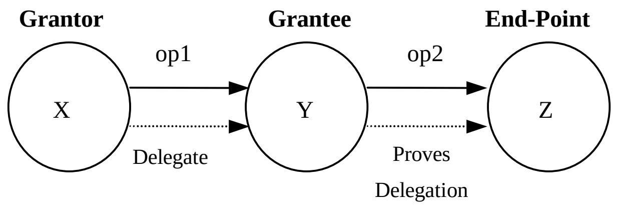
[Figure 2] Role Group and Group Relationship Model

3. Role Delegation Methods

In RBAC, senior role inherits junior role’s permission by virtue of the role hierarchy. But, junior role does not allowed to carry out the permission, which is only granted to the senior or other role groups. When, a senior role fails to operate, junior roles may not continue to perform their jobs when they need the senior role’s permission. Note that, in conventional RBAC, roles are structured hierarchically, and when a role has no permission to certain resources, it should request other roles to access resources on its behalf. Also, role must gain the privileges reserved to other role, which are not granted to him explicitly. For continuous operation, even in the absence of senior role or in the exceptional condition, our delegation scheme makes it possible to grant permissions dynamically in order for roles to carry out their jobs successfully.

Role delegation means, the user delegates his(her) assigned role to the other user in order for him(her) to perform the role’s permissions. We describe a simple scenario in order to explain the necessity of role delegation. Delegation and extended access control policies are illustrated in [5], where delegation is described with domain concepts. A domain is an object that maintains a list of references to other objects, which have been explicitly grouped together for the purpose of management [1][13]. In this paper, apart from the concept of domain, we propose a role delegation by means of inheritance of permissions in senior and junior role-relationship after dividing roles into groups.

Basic delegation concept is illustrated in [Figure 3][1][5]. When X invokes an operation op1 on a target Y, this operation triggers the invocation of op2 on Z. If X has the right to invoke op2 on Z but Y has no right to invoke op2 on Z, then X temporally delegates the necessary access rights to Y in order to invoke op2 on Z. In this case, X acts as the grantor, Y as the grantee, and Z as the end-point [5].



[Figure 3] Basic Delegation Model

For example, in a hospital environment, an intern has no permission to perform surgical operations, which was granted to residents and specialists only. But, sometimes an intern may support surgical operation upon a resident's or a specialist's request. In this case, the specialist who requests the intern to join the surgical operation is the grantor and the intern who supports the surgical operation is the grantee. Consider another example, where a pharmacist has permission to prepare medicine for patients, but a nurse has no such permission. If the pharmacist is absent, a subject who belongs to doctor group can request a nurse to carry out dispense (pnp1) permission. In this case, a nurse plays a pharmacist's role. Nurse is the grantee and doctor is the grantor.

We divide role delegation into active delegation and passive delegation. In active delegation, both the subject who requests the delegation and the grantee are the same subject. In passive delegation, delegation server can delegate the grantor's permission by the other role's request.

3.1 Delegation Server

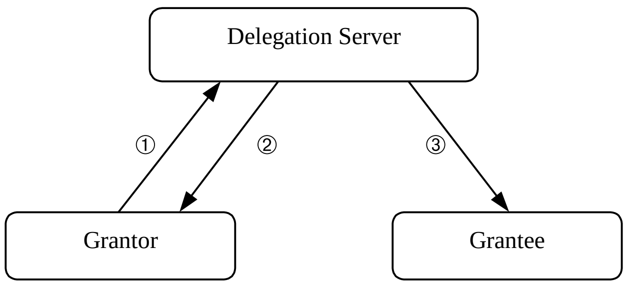
Delegation server makes a decision about whether the delegation process is permitted or not. Server maintains a delegation information for making a decision. In [Table 1], we present some permissions for doctor group, nurse group, and pharmacist group. Permission has mode and exception, which is the delegation information. In [4], mode is divided into rights and duties. Rights, as authorization policies, specify what kinds of activities a subject is permitted (or forbidden) to perform on a target object. Duties are modeled as obligation policies that specify what activities a subject must or must not perform on a target object. Permission's mode has four case a+, a-, o+, and o-. In role delegation methods, permission's mode is very important. Delegation server references the mode for making delegation-permitting decision.

Role has many permissions to perform their jobs. Some permissions are role's rights and some permissions are role's duties. Right permission is delegatable to other roles, but duty permission is not delegatable. Therefore, we assume permission, which has a+ mode, is delegatable to other roles in the same role group. And permission that has a- mode is delegatable just when the exceptional condition is satisfied. But, o+ and o- are duties, which must or must not be performed by role. Therefore, permission, which has o+ or o- mode, is not delegatable. Delegation information consists of permission's mode and exceptional condition, which are represented, in the Role-Permission relationship.

If the role delegation is needed by a subject, the grantor sends delegation request message to the delegation server. Delegation request message consists of delegating role, exception, and grantee, which grants new role by virtue of the role delegation process. Delegation server references delegation information and makes decision about whether delegation can be performed or not after receiving the delegation request message. Then, the delegation server responds to the delegation request message according to the request result. Request result is one of accept or reject. If delegation server finds delegation is needed (accept delegation request), server makes accept message, which contains the grantor, grantee, role, and condition. Server then transmits accept message to the grantor and sends delegated role information to the grantee. Delegated role information consists of role, which was delegated to the grantee, modified role-permission set, and condition. In modified role-permission set, permission's mode is one of a+, a-, or o-. Mode o+ and o- are not delegatable permissions. In modified role-permission set, o+ mode must be changed into o- mode. In case of delegation request reject, server just sends delegation reject message.

3.2 Delegation Protocol

The role delegation protocol can be represented as shown in [Figure 4]. In order to perform role delegation, the grantor transmits the delegation request message to the delegation server (Message ①). Delegation request message contains the grantee, role, and exception. After delegation server makes delegation possibility, server transmits request result (Message ②). The request result is different after the decision of delegation server. If the delegation is permitted, delegation server sends accept message. If the delegation cannot be permitted, delegation server sends reject message. Accept message contains grantor, grantee, role, and condition. Reject message just contains reject delegation. After delegation is permitted, server must transmit delegated role information to the grantee (Message ③) in order for the grantee to perform the delegated role. Delegated role information consists of role, role permission set, and condition. After grantee receives delegated role information, he can perform delegated role, which is mentioned in delegated role information. At this time, grantee can perform permissions, which are specified in role permission set within the delegated role information message.



Message ① : Delegation Request Message
{Grantee, Role, Exception}
Message ② : Request Result
i) Accept : {Grantor, Grantee, Role, Condition}
ii) Reject : {"Delegation request rejected"}
Message ③ : Delegated Role Information
{Role, Role-Permission, Condition}

[Figure 4] Role Delegation Protocol

3.3 Active and Passive Delegation

We model role delegation in two cases, active delegation and passive delegation. Active delegation occurs when a subject actively requests delegation of another role's permission to itself. That is, grantor and grantee are the same subjects. In passive delegation, subject whom request delegation to some other subject rather than itself. That is, grantor and grantee are not the same subjects.

3.3.1 Active Delegation

In active delegation, there exists an exceptional field with a- mode of the Role-Permission relationship [Table 1]. If the exceptional requirements are satisfied, the grantor demands the delegation server for the delegation of the appropriate role to itself. In this case, grantor and grantee in [Figure 4] are the same subjects.

Scenario 1] Nurse Delegation to himself

In nurse group, the permission of a preparation of medicine (np3) is a- mode in Role-Permission relationship. Thus nurse group has no right to prepare medicine in a normal situation. Permission np3 has exceptional condition emergency. That is, when emergency condition occurs, nurse can perform preparation of medicine. If the exception condition (emergency) is satisfied, nurse request role delegation to delegation server about preparation of medicine, which is originally pharmacist's permission. Nurse transmits delegation request message {nurse, pharmacist, emergency} to the delegation server. Delegation server judges delegation request and makes accept request result message. Request result message is {nurse, nurse, pharmacist, emergency}. After transmitting result message, server transmits delegated role information. In this active delegation, grantee has a- permission of delegated role. Thus, the server simply changes np3 permission's mode into a+, and transmits {nurse, nurse-changed np3, emergency} message to grantee (nurse). Also after the delegation process, in emergency condition, chief nurse has changed np3 permission. Chief nurse inherits nurse's permission because nurse is a junior role in nurse role group.

3.3.2 Passive Delegation

In passive delegation, grantor and grantee are not the same subjects. Grantee can carry out a new acquired role by the delegation process. DAC is free for delegating one's permission to other users, but in RBAC, since permission is granted to roles but not to users, only system administrator can change Role-Permission relationship. Sometimes, one role delegates his permission to the other role. Or, the grantor's permission is delegated to the grantee by the other role's request. Passive delegation can be triggered by any role in the system.

Scenario 2] Resident delegates his role to intern

Take a resident as an example. Resident has dp3, and resident supports surgical operation upon a specialist's request. But sometimes the resident can not perform that permission, and he wants that permission to be performed by an intern. At that time, the resident decides to delegate his role to the intern in order to carry out the specialist's request.

Resident makes delegation request message, {intern, resident, -}, for delegating his role to intern. Then delegation server makes decision about whether delegation can happen. If delegation server decides to allow this delegation, then delegation server transmits accept request result message, {resident, intern, resident, -}. Delegation server must send delegated role information {resident, resident's permission set, -} to intern. Intern, who is the grantee in the delegation process, can perform resident's permission, which is mentioned in resident's permission set.

Scenario 3] Doctor delegates pharmacist's role to nurse

Take np2 permission as another example for passive delegation. This permission means that nurse has no authority about preparation of medicine, except in the emergency condition. If one member of the doctor group needs nurse's preparation of medicine in emergency condition, he requests to delegate the pharmacist's role to nurse. Grantor (member of a doctor group : ex, intern) makes delegation request message, {nurse, pharmacist, emergency}. After delegation server receives request message it defines whether delegation is permitted or not. If delegation can be permitted, server transmits accept request result message, {intern, nurse, pharmacist, emergency}, to the grantor. Delegated role information must transmit to the nurse, to whom the pharmacist's role is granted. Delegated role information is the same, which is used in scenario 1.

Scenario 4] Nurse delegates specialist's role to resident

Take a resident as an example. A resident does not have permissions of reading and fixing charts written by interns. When a nurse finds problems in chart written by intern, he (she) has to demand fixing the specialist who owns the permission dp1. If the specialist does not exist, the role higher than the intern is called on to judge the correctness of the chart. The nurse delivers message {resident, specialist, no specialist} to the delegation server. After deciding the possibility of delegation, the delegation server lets the nurse know the result with request result message. In this scenario, request result message is {nurse, resident, specialist, no specialist}. Delegated role information message is {specialist, specialist-permission set, no specialist}. Note that, specialist-permission set is not the same in original Role-Permission relationship. Dp2 must be changed similar to {dp2, o-, specialist, {chief of surgical operation}, patient, -,-}. When delegation process occurs, mode o+ must be changed into o-. This is because, duty permission can not be performed by a delegated role.

Scenario 5] Delegation reject

Intern has o- mode permission, dp6. In this case, intern can not perform {chief of surgical operation}, which was granted to specialist. If no specialist exists, nurse can make the decision on surgical operation for patients. Nurse makes delegation request message, {intern, specialist, no specialist}, and transmits it to delegation server. Delegation server makes reject message, because dp6 permission contains mode o-. Also, {chief of surgical operation} is just granted to specialist by dp1, with o+ mode. Delegation server transmits request result message, {"Delegation request rejected"}. Delegation server rejects delegation process.

4. Conclusion and Future Works

We have presented role-based access control (RBAC) concept that is in the spotlight recently, and have defined its simplified model. RBAC model shows the standardized access control of complicated organization’s resources.

In RBAC, permission is assigned not to user but to role. In case of changing access control policy, Role-Permission relationship, not User-Permission relationship, is to be revised. Therefore, RBAC can deal with changing access control policy more flexibly.

In RBAC, senior role has junior role’s permission by virtue of role hierarchy. But, junior role can not perform the permission, which is granted to the senior or other role groups only. If some kinds of role fail to perform their permission, junior roles or other roles also cannot perform their jobs. For continuous operation, even in the absence of senior role or in the exceptional condition, our delegation scheme allows dynamically granting roles to perform their jobs successfully. Inheritances of permissions in role hierarchies are static. In order to tackle this problem, we propose a role delegation in which appropriate roles and permissions can be performed temporarily.

Role delegation server makes a decision about whether delegation process is permitted or not. Server maintains a delegation information for making decisions. Delegation information is maintained in Role-Permission relationship. If the role delegation is needed, grantor makes and sends delegation request message to delegation server. Delegation server makes decision about delegation is performed or not after retrieving delegation information. Also, delegation server transmits request result to grantor and transfer delegated role information for grantee.

Role delegation is classified into passive delegation and active delegation. The former is that other role request delegation, and the latter is that grantee and delegation requester are the same.

Our delegation methods propose a role delegation to other roles or itself. In role delegation, obligation is not delegated to other roles. Only authorization is delegated to other roles. In future work, we address delegation granularity. This is needed to delegate dedicated permissions to other roles instead of the whole role. This can be done through specifying constraints for delegation information.

5. REFERENCES

[1] E. C. Lupu, D. A. Marriott, M. S. Sloman, and N. Yialelis, “A Policy Based Role Framework for Access Control”, First ACM/NIST Role Based Access Control Workshop, Dec, 1995.

[2] Department of Defense(USA), Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200-78-STD, DoD, 1985.

[3] L. Giuri, “Role-Based Access Control in Java”, 3rd ACM Role-Based Access Control Workshop, 1998.

[4] E. C. Lupu, M. S. Sloman, “A Policy Based Role Object Model”, Proceeding of IEEE EDOC'97, Oct, 1997.

[5] N. Yialelis, M. S. Sloman, “A Security Framework Supporting Domain Based Access Control in Distributed Systems”, ISOC Symposium on Network and Distributed System Security(SNDSS96), Feb 1996.

[6] David F. Ferraiolo and Richard Kuhn, “Role-based access control”, Proceedings of the 15th NIST-NSA National computer security conference, 1992.

[7] Ravi S. Sandhu, Edward J.Coyne, Hal L. Feinstein and Charles E. Youman, “Role-Based Access Control Models”, IEEE computer, Volume 29, number 2, Feb 1996.

[8] David F. Ferraiolo, J. Cugini and Richard Kuhn, “Role-Based Access Control: Features and Motivations”, National Institute of standards and technology, 1995.

[9] J. Barkley, “RBAC in Health Care”, 1995. http://hissa.ncsl.nist.gov/rbac/

[10] C. Goh, A. Baldwin, “Towards a more Complete Model of Role”, 3rd ACM Role-Based Access Control Workshop, 1998.

[11] B, Lampson, M, Abadi, and R, Needham, “A Logic of Authentication”, ACM Transaction on Computer System, Vol. 8(1), 1990.

[12] Fang Chen and R, S Sandhu. “Constraints for Role-Based Access Control”, ACM RBAC Workshop. MD. 1996.

[13] M. Sloman. “Policy Driven Management for Distributed System”, Journal of Network and System Management, Vol.2(4), 1994.

[14] Eun-Hong Cheon and Dong-kyu Kim, “A Model and Constraints for Role Based Access Control(RBAC)”, Proceeding of The 24th KISS Fall Conference, V.24. n.2, 1997.

[15] Nicholas Yialelis, Emil Lupu and Morris Sloman, “Role-based Security for Distributed Object Systems”, Proceedins of the IEEE Fifth WET ICE, 1996.

[16] NIST ITL Bulletins, “An Introduction to Role-Based Access Control”, 1995.

[17] D. Marriot and M. Sloman, “Management Policy Service for Distributed Systems”, IEEE Third Int. Workshop on Servics in Distributed and Networked Environments(SDNE’96), Macau, June 1996