A Novel Approach to Role-Based Access Control*

Song-hwa Chae[1], Wonil Kim[1], and Dong-kyoo Kim[2]

[1]The Graduate School of Information and Communication, Ajou University,
Suwon, 442-749, Republic of Korea
{portula,wikim}@ajou.ac.kr
[2]College of Information and Computer Engineering, Ajou University,
Suwon, 442-749, Republic of Korea
dkkim@ajou.ac.kr

Abstract. With the rapid growth of distributed and network systems, sharing resources among many users become more common. As a result of that, we encounter with new problems concerning security and privacy on the shared resources. An access control mechanism such as role-based access control (RBAC) is one of the solutions to cope with these problems. RBAC is an efficient access control mechanism for organization data with role and permission management. In this paper, we propose a new implementation method for RBAC, which uses neural networks instead of tables. By employing neural network, it has advantages of not using multiple storages for role-permission tables and extra mutual exclusive data tables. It also reduces access time for requested role and permission sets.

1 Introduction

As resource sharing among many users becomes more prevalent, we need to apply proper access mechanism on the shared resources. System administrators and software developers need to manage resources efficiently and ensure proper access control of them. An access control mechanism such as role-based access control is widely used method to solve these problems.

There have been many researches on access control mechanism in information security. Access control mechanism is categorized into three areas, such as mandatory access control (MAC), discretionary access control (DAC) and role-based access control (RBAC). MAC is suitable for military system, in which data and users have their own classification and clearance levels respectively. DAC is another access control method on objects with user and group identifications. RBAC has emerged as a widely acceptable alternative to classical MAC and DAC [2][5]. It can be used in various computer systems.

In this paper, we propose a new implementation method for access control mechanism. The main idea is employing neural network for role-based access control mechanism instead of fixed tables. It enables to eliminate the search time of relation tables and easily detects mutual exclusive roles.

* This study was supported by the Brain Korea 21 Project in 2003.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1060–1068, 2003.

Chapter 2 explains the basic concept of RBAC and neural network, chapter 3 describes the proposed RBAC system using neural network, and chapter 4 evaluates the proposed system and chapter 5 concludes with future works.

2 Background

2.1 RBAC

RBAC is a very popular mechanism widely used in both research and industry. It does not allow users to be directly associated with permissions, instead each user can have several roles and each role can have multiple permissions. A relationship between users, roles and permissions is shown in Fig1.

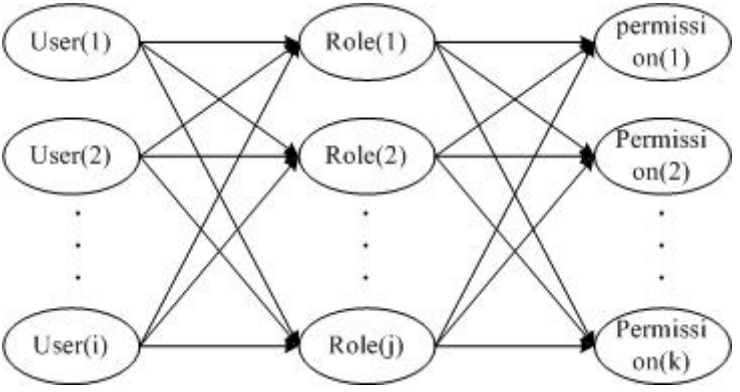

Fig. 1. Relationship among user, role and permission

Each group can be represented as a set of user U, a set of role R, and a set of permission P.

- $U = \{u_1, u_2, \cdots, u_i\}$
- $R = \{r_1, r_2, \cdots, r_j\}$
- $P = \{p_1, p_2, \cdots, p_k\}$

Two different types of association must be managed by the system; one is the association between user and role, the other is the association between role and permission [6]. It is characterized as user-role (UR)[1] and role-permission (RP) relationship[2].

- $UR = \{u \in U, r \in R \mid u \rightarrow 2^{|r|}\}$
- $RP = \{r \in R, p \in P \mid r \rightarrow 2^{|p|}\}$

[1] UR is role assignment function that assigns roles to users.

[2] RP is permission assignment function that assigns permissions to roles. Both UR and PR are represented by bits. It means active or inactive.

Consequently, in order to have proper management, the system needs to maintain two separate association tables. General RBAC implementation uses relation tables as shown in Table 1 and Table 2.

Table 1. User and role example table.

	r_1	r_2	⋯	r_i
u_1	1	0	⋯	1
u_2	1	1	⋯	1
⋯	⋯	⋯	⋯	⋯
u_j	0	0	⋯	1

Table 2. Role and permission example table.

	p_1	p_2	⋯	p_k
r_1	1	0	⋯	1
r_2	1	1	⋯	1
⋯	⋯	⋯	⋯	⋯
r_i	0	0	⋯	1

2.2 Mutual Exclusive Role in RBAC

Conflicts of interest in a role-based system may arise as a result of a user gaining authorization for permissions associated with conflicting roles [4]. For example, if one role requests expenditures and another role approves them, the system must prohibit the same user from being assigned or active to both roles. In order to solve these conflict problems, there have been many researches since middle of 1990's. Finally, National Institute of Standards and Technology (NIST) proposed two Separation of Duty (SOD) reference models in 2001; Static Separation of Duty (SSD) and Dynamic Separation of Duty (DSD). To implement these reference models, the system must have extra tables to protect against activating mutual exclusive roles. According to the NIST reference models, SSD should check the mutual exclusive role in every role assignment, and the role-permission in every user's session request [Fig2]. DSD should check all the three relations for every user's session request [Fig3]. SSD is rarely used in practice and DSD imposes a lot of overload in a system with many users due to excessive table accessing time. In fact, general processes do not always use mutual exclusive roles.

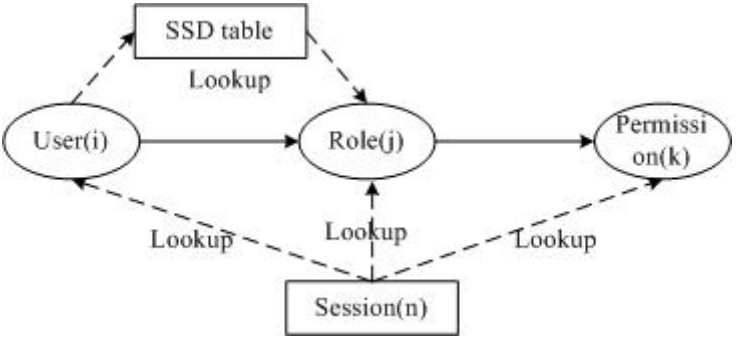

Fig. 2. SSD within RBAC

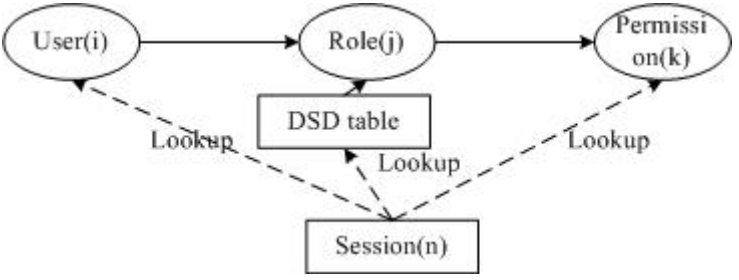

Fig. 3. DSD within RBAC

2.3 Neural Network

Neural network is modeling of human brain's neuron processing. The proposed system employs neural network instead of relation tables. It is trained using backpropagation algorithm.

Backpropagation (BP) algorithm [1] is one of the method of neural network learning. It learns the optimal mapping between input and desired output data by iterative weight update. In this paper, two neural networks are used; one learns the mapping from roles to permissions, and the other learns the reverse mapping between them, from permissions to roles.

3 RBAC Using Neural Network

When a system employs RBAC, it needs at least 3 fixed tables for a session, such as user-role table, role-permission table and mutual excusive role table. It requires extra storage as well as checking time for both role-permission and mutual exclusive role tables. To cope with these problems, we propose an effective implementation method

of RBAC. The proposed RBAC system uses neural networks instead of fixed RBAC relation tables. By employing neural network in RBAC, the system can check the user's permissions in each session without using relation tables. This method does not only reduce access time for authorization but also prevent a user from being activated with mutual exclusive permission. It enables the system effectively applied to DSD in RBAC with many users.

3.1 System Architecture

In the proposed system, roles, permissions and mutual exclusive roles are defined by security administrator as in other RBAC systems. The neural network is trained to learn the relations among them. When a particular user logs into a system, the system is given the user's role set. The system produces the user's whole permission set by using the neural network. The permission provided by neural network is used for authorization. It is assumed that roles, permissions and their associations are static information, whereas associations between user and role are dynamically changed. User can have several roles and role may have multiple permissions too. The process of this system consists of three phases: 1) neural network learning phase, 2) role assignment phase, 3) permission extraction phase. Depending on whether a user's role set contains mutual exclusive roles or not, two cases are considered in system processing.

3.2 Non-mutual Exclusive Role Case

This case is SSD in RBAC. Users do not contain mutual exclusive roles. Thus, the system checks only user's role and permission.

1. Neural Network Learning Phase
   In the learning phase, the training data is the role-permission relation provided by the system administrator. Each role and permission is represented by input and output respectively. The value of input (role) and output (permission) is either '1' (active/permit) or '0' (non-active/denial). Since the proposed system should be able to accommodate hierarchical RBAC, the high level role may contain the low level permissions too. After the neural network learns the relation, the system should be able to respond user's permissions for access control.
2. Role Assignment Phase
   In this system, a user is defined as a human being and a role is a job function within the context of an organization [4]. Therefore, the system can assign multiple roles to the user according to the required job. The association of the user and the role can be changed dynamically. Our RBAC system can successfully respond to these changes. It produces the proper set of permissions dynamically even though the user-role association changes.
3. Permission Extraction Phase
   When the user tries to access data, the system makes decisions such as permit or denial. The proposed system makes the decision using the user's permission set. This permission set is generated by the neural network in the user logging phase.

This set has the user's whole permissions. Since all the permissions do not contain any mutual exclusive role, mutual exclusive resolution is not necessary.

3.3 Mutual Exclusive Role Case

This case is DSD in RBAC. The user can have multiple roles, among which may contain mutual exclusive roles, such as requesting expense and approving that. These two permissions come from two different roles. The main point of this process is to reduce the role set, so that the reduced role set does not have any mutual exclusive permission. With this process, the neural network will be able to produce the reduced permission set according the reduced role set. The process of producing the reduced permission set is as follows.

1. Neural Network Learning Phase
   In this case, we represent permissions with three types of values; '0','1', and '0.5'. The '0' and '1' has the same meaning as defined in non-mutual exclusive role case. Especially, '0.5' means mutual exclusive permission. An example of this is shown in A.1. If there is a high level role containing low level mutual exclusive permission, that node (permission) is also set to '0.5'. In this case, second neural network is employed to produce the reduced role set. It will lean the relation between permissions and roles. Each permission and role will be represented by input and output respectively. An example of this is shown in A.2.
2. Role Assignment Phase
   Same as in non-mutual exclusive case.
3. Permission Extraction Phase
   The DSD system must decide permit or denial on the given request in every session. The proposed system makes the decision using the user's permission set. This permission set is generated by the neural network in the user logging phase. This set has the user's whole permissions including mutual exclusive permission. The process of making the whole permission set is similar to non-mutual exclusive case. When the user tries to access data defined as mutual exclusive permission, the system recognizes mutual exclusion case using permission's value, which is approximately '0.5'. In this case, the system should make a least user's permission set for the session. This permission set is a reduced set which does not contain mutual exclusive permission. Second neural network is employed to produce reduced role set. In order to limit user's permission, it changed a bit to '1' in user's permission set and others to '0'. This changed user's permission set is used to produce the limited role set. The limited role set is used to produce the reduced permission set using the first neural network for this session. After this process, the user has reduced permission set which is not containing mutual exclusive permission. In any session, if the request permission is found to be the mutual exclusive permission, the system does the same process recursively. After the session, it will return to the previous permission set. This process protects from executing mutual exclusive permission in any given session.

4 Evaluation

The neural network was trained and evaluated on a prototype organization [4], such as a small company in Fig 4. It had 2 work parties, 10 roles and 6 mutual exclusive roles. This system could accommodate unlimited number of users. The evaluation result showed that the proposed system detected mutual exclusive role activation and produced the user's permission set. The role-permission relationship of Fig 4 is represented in A.1.

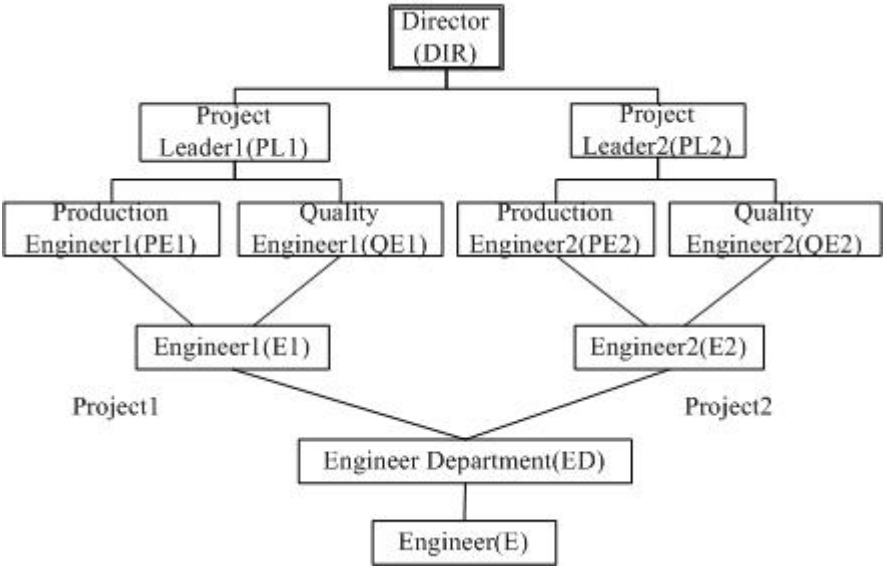

Fig. 4. Example of an organization

We assumed that PL1 and PL2 had mutual exclusive roles. PE and QE had mutual exclusive roles too. Therefore, the permissions of these roles are set to '0.5'.

We trained two neural networks using A.1 and A.2 respectively, with learning and the momentum rates are 0.001. The proposed neural network had one hidden layer (30 nodes) and used sigmoid functions for hidden and output nodes. The five sample users and their role set (representations) are given below;

- User1 : {PE1, PE2} : (0100000100)
- User2 : {PE1, QE2} : (0100001000)
- User3 : {PE1} : (0100000000)
- User4 : {PL1, PE2} : (0001000100)
- User5 : {DIR} : (0000100000)

The system produces user's whole permission set using first neural network. If the user tries to access any mutual exclusive permission, it produces reduced role set using second neural network. For example, user5 is assigned as DIR, the first neural network will produced the whole permission set of {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10} with access values between 0.0 and 1.0. If DIR try to access any mutual

exclusive permission, such as p9, the system recognize it's mutual exclusive case and produce the reduced role PE1 using second neural network. As a result, the role of user5 which was originally defined as a DIR, has reduced to PE1 in this particular session and the permission sets as {p4, p7, p9} accordingly. The whole role and permission set and reduced role and permission set of User1~ User5 are shown in Table3.

Table 3. Whole role set and reduced role set

Case	Whole Role Set	Whole permission Set	Requested Job(permission)	Reduced Role Set	Reduce Permission Set
user1	PE1,PE2	{p2,p4,p7,p9,p10}	p9	PE1	{p4,p7,p9}
user2	PE1,QE2	{p2,p4,p7,p8,p9}	p8	QE2	{p2,p4,p8}
user3	PE1	{p4,p7,p9}	p9	PE1	{p4,p7,p9}
user4	PL1,PE2	{p1,p2,p3,p4,p7,p9,p10}	p10	PE2	{p2,p4,p10}
user5	DIR	{p1,p2,p3,p4,p5,p6,p7,p8,p9,p10}	p9	PE1	{p4,p7,p9}

5 Conclusion and Future Works

RBAC is efficiency access control method for organization data with role and privilege management. It constructs with role, privilege, user, and session, is implement with each relation tables. In this paper, we proposed a new implementation method for RBAC. The proposed methods can be applied dynamically with user's role change and has advantages of not using multiple storages for role-permission tables and extra mutual exclusive data tables. It also reduces access time by eliminating excessive table search for mutual exclusive roles. This method can be easily extended to various access control mechanisms.

References

1. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. Nature, 323, (1986) 533–536
2. E.H.Choun, A Model and administration of Role Based Privileges Enforcing Separation of Duty. Ph.D. Dissertation, Ajou University(1998)
3. K.Mehrotra, C.K.Mohan,S.Ranka, Elements of Artificial Neural Networks, MIT Press (1997)
4. D.F.Ferraiolo, R.Sandhu, E.Gavrila, D.R.Kuhn, R.Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and System Security, Vol4, No3 (2001) 224–274
5. G.Ahn, R.Sandhu, Role-Based Authorization Constraints Specification, ACM Transactions on Information and System Security, Vol3, No4,207–226 (2000)
6. D.FFerraiolo, J.F.Barkley, D.R. Kuhn, A Role-Based Access Control Model and Reference implementation Within a Corporate Intranet, ACM Transactions on Information and System Security, Vol2, No1 (1999) 34–64

Appendix

A.1 Input and Output Data for First Neural Network Training

	Input data	Out data									
	Role	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
ED	0000000001	0	0	0	1	0	0	0	0	0	0
E1	1000000000	0	0	0	1	0	0	1	0	0	0
E2	0000000010	0	1	0	1	0	0	0	0	0	0
PE1	0100000000	0	0	0	1	0	0	1	0	1	0
PE2	0000000100	0	1	0	1	0	0	0	0	0	1
QE1	0010000000	1	0	0	1	0	0	1	0	0	0
QE2	0000001000	0	1	0	1	0	0	0	1	0	0
PL1	0001000000	0.5	0	1	1	0	0	1	0	0.5	0
PL2	0000010000	0	1	0	1	0	1	0	0.5	0	0.5
DIR	0000100000	0.5	1	0.5	1	1	0.5	1	0.5	0.5	0.5

A.2 Input and Output Data for Second Neural Network Training

	Input data										Out data
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	Role
ED	0	0	0	1	0	0	0	0	0	0	0000000001
E1	0	0	0	1	0	0	1	0	0	0	1000000000
E2	0	1	0	1	0	0	0	0	0	0	0000000010
PE1	0	0	0	1	0	0	1	0	1	0	0100000000
PE2	0	1	0	1	0	0	0	0	0	1	0000000100
QE1	1	0	0	1	0	0	1	0	0	0	0010000000
QE2	0	1	0	1	0	0	0	1	0	0	0000001000
PL1	0	0	1	1	0	0	1	0	0	0	0001000000
PL2	0	1	0	1	0	1	0	0	0	0	0000010000
DIR	0	1	0	1	1	0	1	0	0	0	0000100000