A COMBINED FINE-GRAINED AND ROLE-BASED ACCESS CONTROL MECHANISM

By

HONGI CHANDRA TJAN

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

2001

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the degree of
MASTER OF SCIENCE
May 2006

A COMBINED FINE-GRAINED AND ROLE-BASED
ACCESS CONTROL MECHANISM

Approved By:

Dr. Johnson P. Thomas
Thesis Adviser

Dr. George E. Hedrick

Dr. Debao Chen

Dr. Gordon Emslie
Dean of the Graduate College

TABLE OF CONTENTS

Chapter **Page**

TABLE OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

Chapter 1

Introduction

Companies and enterprises need a reliable and secure system to manage and analyze their data. Researchers have developed several kinds of secure systems for data access. In particular, Fine-Grained Access Control (FGAC) system and Role-Based Access Control (RBAC) are widely used secure systems [Andrei, 2005]. The FGAC system determines access control rights based on individual data. The access rights of every user allowed to access the data are stored inside the data itself [Damiani, 2002]. In RBAC, access rights are defined based on the position or role of the user [Zhang, 2003]. There may be multiple users with the same role. These multiple users may have access to the group or collection of data.

In FGAC, the access rights of a user are stored in the data itself [Damiani, 2002]. Hence, if multiple users have access to the same data, the data must be replicated multiple times. This results in a complex system which does not scale well [Ahn, 2000]. The RBAC system has a simpler environment. Since all the users are grouped based on categories, the access right to data is based on the user's category. The problem with the RBAC system is the absence of identity of the user. Since all the users are recognized by their categories, the real identity of the user is lost. All activities of the user in the system

are based on their categories, and the user's real identity is therefore unknown. Therefore, a RBAC system results in a less secure system.

Combining both the FGAC and RBAC system can solve the problem of complexity and lack of security. The main objective of combining FGAC and RBAC is to inject the users' identity into the RBAC system while reducing the amount of data replication involved. In the combined FGAC-RBAC systems, all the users in the system are categorized into several groups, and at the same time, the users still retain their identity. When the user accesses the data, the access permission is based on the user's category. All activities of the users are based on the users' identity and category. The combined system is much simpler because data is not replicated.

Graphs are used to present the combined system. Graphs formalism is used to define the process of combining the two systems. The following figure is an example of a graph:

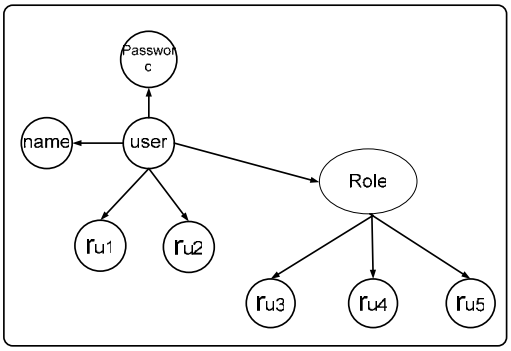

Figure 1. Example of Graph for Combined System

Automata, and grammar are used in addition to the graph. In our approach, the process of accessing data is represented as a number of transitions from one state to another. An automaton is used to capture the transition between states and the corresponding inputs and outputs in our system. Furthermore, the automaton is formally

represented using grammatical representation. Properties of the combined system are proven.

A definition for the terms used in the thesis is presented in chapter 2. A review of FGAC and RBAC systems is presented in chapter 3. In chapter 4, we propose our solution to the combined access control system. In chapter 5, we show a variation of the combined FGRBAC system such as the FGAC priority over RBAC in the system in single user with single role as explained in chapter 5. Partial results of the simulation are shown in chapter 6. The simulation in this chapter shows the differences of the information load stored inside FGAC, RBAC, and FGRBAC systems. An algorithm, a DFA, and a grammar are used to support the FGRBAC system shown in chapter 7. The results are summarized in Chapter 8.

Chapter 2

Definitions

In this chapter, terms used in this thesis are defined.

CFG. Context-free-grammar, a formal grammar that contains four important components in the grammatical description of a language, that is: G = (*V,T,P,S*), where:

V → Variables, which is also called *nonterminal* or *syntactic categories*.

T → Terminal or terminal symbols, which is a finite set of symbols that form the stings of the language being defined.

P → Productions, which is a finite set of rules that represent the recursive definition of a language. Each production consists of:

- A variable that is being defined by the production. This variable is often called the head of the production.
- The production symbol '→'
- A string of zero or more terminals and variables. This string, called the body of the productions, represents one way to form stings in the language of the variable of the head.

S → Start symbol, which is one of the variables, represents the language being defined.

Collective rule. A rule that corresponds to the user's role, not the user itself (rule in RBAC).

Data Rules. rules that belong to the data in the system, which define the access right of users. (Defines which user can read or write the data).

Delegation. The process of assigning a role to a new user by another user who is more senior or has a higher ranking in the system.

DFA. Deterministic Finite Automata, is a quintuple M = (K, Σ, δ, s, F) where

K is a finite set of states,

Σ is an alphabet,

$\delta \in K$ is the initial state,

F a subset of K is the set of final states, and

δ is a function from K cross Σ to K.

FGAC. Fine-grained access control, a system that allows the definition and enforcement of access restrictions for individual users directly on the structure and content of the documents.

Individual rule. The right of every user that responds directly to the user itself without going through the user's role.

RBAC. Role-based access control, a security system that is based on the roles of the user or position of the user inside the system.

Revocation. The opposite of delegation, that is, the process whereby an active entity is withdrawn from a distributed environment by another entity that is superior to the eliminated entity.

Rule. Defines the access right of the user to the data. Categorized as **Individual Rule, Collective Rule,** and **Data Rule.**

Rule based access control. Access control system that is based on the rules where all access permission is controlled by rule comparison (between rules that belong to data and rules that belong to either the user or the role).

Chapter 3

Literature Review

Fine-grained access control was introduced by E. Damiani in his paper, "A fine-grained access control system for XML documents." [Damiani, 2002]. Prior to fine-grained systems, file-system security depended on the file itself (file-level systems) [Ravi, 2000]. A fine-grained security system protects the entire system at a fine grain so that the system can be accessed only by an authorized user. In file-level systems, everyone could access the system not simply authorized users. Thus, the fine-grained security system introduced by Damiani was better than the previous system that implemented security at a file-level.

The Fine-Grained Access Control (FGAC) system is based on "individual rules" [Damiani, 2002]. Individual rules are the rules associated with or corresponding directly to the user. These individual rules describe clearly what data the user can access and what data the user cannot access. The following graph is an example of a FGAC system [Koch, 2002]:

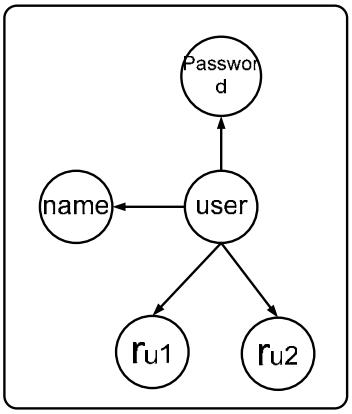

Figure 2. Example of FGAC

In Figure 2, *ru1* and ru2 are individual rules that describe the rights of the user. The definition of the fine-grained access control in Figure 2 is as follows:

The components of the system are:

User: The person who is accessing the system.

Name: User's identification, which is unique or specific for every user.

Password: User's security code, which verifies the identification of the user

Individual rule: The rule(s) that define the rights of the user in the system.

- Each vertex represents the following:

  User: us = a User

  Name: n = name for user

  Password: p = password for user

  An individual rule k: IR_k

  The set of individual rules is: iru = $\{IR_1, IR_2, \ldots, IR_m\}$ where IR_i is individual rule i

  The set of individual rules for a user i: $iru_i \subseteq iru$

- The FGAC access control system graph is defined as follows

G = {VF, EF}

Where

$$VF = \{\text{us}, \text{n}, \text{p}\} \cup iru$$

$$EF = \{\{(\text{us}, \text{n}), (\text{us}, \text{p})\} \cup (\bigcup_{i=1}^{m} \{(us, IR_i)\}$$

In the FGAC system, a user logs in with his username and password. The information pertaining to a user (such as his username and password) are also stored in the data. Other security related information pertaining to data such as the access rights of the user in relation to that particular data are also stored as part of the data. These are stored as individual rules within the data. Information pertaining to a user such as his username, passwords, and the data he can access with corresponding access rights, size of e-mail permitted for the user, etc. is also stored separately as individual rules associated with that user in the FGAC system. When the user wants to access a datum, the datum will match the user's information. This is stored as individual rules within the data along with the separate individual rules that store the user's information to verify the rights of the user [Wainer, 2005]. The FGAC system is, therefore, a very complex system because of its individuality. When added into the system, a new user is assigned individual rules. Some of them may be new rules, whereas others may be existing rules when rules such as access rights already exist. However, the associated parameters such as the name of the user must be added.

The data in an FGAC system needs to keep all the users' information. If there is a new user added to the system, the authentication information of the database needs to be updated, otherwise, he or she will not be able to access the data. Therefore, this kind of

system is complicated. If the number of users and data is permanent and small, an FGAC system can be used. If a system is flexible (frequently updated) and large, an FGAC system incurs a large overhead.

Role-Based Access Control system is the other commonly used security system [Ahn, 2000]. In the Role-Based Access Control system (RBAC), users have access based on the role(s) of the users in the system. This means every user who is allowed to access the system has his/her roles, and, once the user is inside the system, the identity of the user is ignored. The only information that matters is the user's role.

The RBAC system is also based on rules [Ahn, 2000]. The rules in RBAC are recognized as collective rules or group rules. Collective rules correspond to the user's role. These collective rules define what data a role is allowed to access. The following figure is an example of an RBAC system [Koch, 2002]:

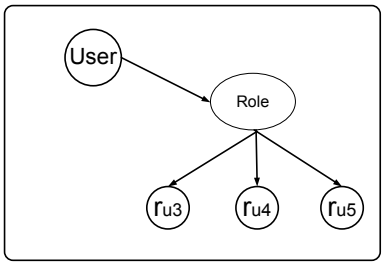

Figure 3. Example of RBAC with Collective Rules

In Figure 3, ru3, ru4, and ru5 illustrate collective rules, which describe what data the user's role can or cannot access.

The definition of the role-based access control in Figure 3 is as follows.

- Explanation of the vertices:

  Role: The position of the user inside the system.

  Collective or group rule: The rule(s) define the right of the role in the system

- Each vertex represents the following:

  Role: roi represents Role for user i

  A collective rule j:

  The set of collective rules is: cru = $\{CR_1, CR_2, \ldots, CR_n\}$- where CR_i is collective rule i

  The set of collective rules for a user i: $cru_i \subseteq cru$

- The RBAC access control system graph is defined as follows

  G = {VR, ER}

  Where

$$VR = \{\text{ro}\} \cup cru$$

$$ER = \bigcup_{i=1}^{n} \{(ro, CR_i)\}$$

In the RBAC system, collective rules that specify the roles of a user are stored in the system. These new rules are also stored in the data specifying which roles can access the data. When a user wants to access data, the system will match the collective rules stored in the system with those stored in the data to verify the rights of the user. When the data in the system has been compromised by an attacker, the role of the user who accessed the data will be recognized, but not the identity of the individual himself. Therefore the RBAC system is insecure.

The database in the RBAC system must retain all the information about the user's role [Delis, 2005]. The RBAC system creates a very simple process if there is a new user added to the system. The user can be assigned the appropriate roles without creating new rules. Not all the data has to be updated as in a FGAC system. However, the security is

weak compared to the FGAC system. Therefore, the RBAC system can be used in any system where data is not critical. If the database of the company is critical, the RBAC system provides less security than FBAC.

The recent research done by Wainer and Kumar [Wainer, 2005] shows a simple delegation and revocation process in the RBAC system. The delegation method in Wainer's paper is the user-to-user delegation method. [Zhang, 2003] The method introduced involves the delegation and revocation of certain rights as opposed to Zhang's method, which delegates and revokes at the role level. Revocation and delegation at the role level means the revocation and delegation of all access rights of the user.

There are pros and cons with both Wainer's and Zhang's system. If the system only requires a simple delegation or revocation of certain rights, then this system is the best choice. On the other hand, if all the user's rights need to be delegated or revoked, then this system is not the best choice.

The delegation and revocation system [Wainer, 2005] is good for certain access rights because all the rights are delegated or revoked one by one. This system is bad for delegation and revocation of a large number of access rights for the same reason, because it cannot delegate or revoke all the access rights of multiple users at one time.

Chapter 4

Steps for Accessing the Combined Access Control system

Security systems such as Fine-Grained Access Control (FGAC) and Role-Based Access Control (RBAC) have several problems such as insufficient security and complexity. We propose to integrate the FGAC into a RBAC system by adding the users' identities to the RBAC system instead of eliminating them as in the original RBAC system. Hence, our proposed system provides the security of FBAC and the simplicity of RBAC.

In the combined system, there are two kinds of accounts activated when a user signs in to the system. One of the accounts is a fine-grained system account, where the user's personal identity is stored. The other is a role-based system account, where the role or position of the user in the system is stored. Further explanation is given in Figure 4.

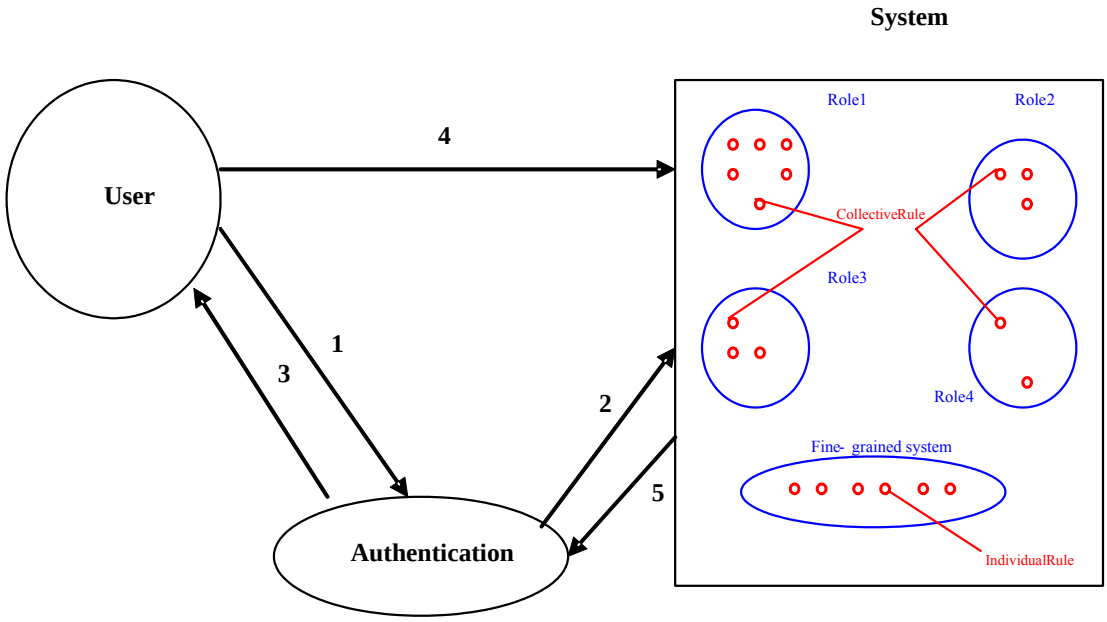

Figure 4. Combination of Fine-grained and RBAC

The steps involved in a combined system are shown in Figure 4:

1. The user enters identification and password.
2. If the user's identification and password is authenticated, the user is directed to the system and assigned to his or her role in the system. In this step, two accounts are activated; the first is the user identification (fine-grained system); the second is the user's role(s) in the system (role-based system).
3. If the user's permission is denied, then the user is not allowed to enter the system and is informed of his or her rejection.
4. After the user is granted access, he or she can use the system directly without going through the authentication stage.
5. Whenever he or she wants to store edited data, the user is requested to re-enter his or her information and password.

The difference between role-based access control and the combination of Fine-Grained and Role-Based Access Control (FGRBAC) is defined in step 4. In the RBAC, only one account is activated, that is, the user's role. Figure 4 shows, the combination of FGAC and RBAC in the system.

In the RBAC system, once the user logs in into the system, the user's identification is not checked again. This situation happens after step 4 in Figure 4. When the user needs to store the edited data, he does not need to verify his ID or password. This means that step 5 in Figure 4 does not exist in the RBAC system.

In the FGAC system, the entire user's activity has to pass the individual rules stored inside the system itself. The individual rules are the rules inside the fine grained system in Figure 4. Once the user is permitted into the system, all the files that are allowed to be accessed by the user, which include read, write, delete, store, etc., are defined by individual rules. This situation can create a complicated and huge number of individual rules.

The integration of the FGAC into the RBAC system is to keep the user's personal identity in the system for security reasons. For example, if there is a problem in the data, the system can trace the identity of the users who accessed the data. If the system does not contain the FGAC system, it cannot trace the identities of the users who access the data.

Chapter 5

FGAC-RBAC combined System

In this thesis, we combine FGAC and RBAC by integrating FGAC into RBAC or adding the identity of the user in the RBAC system.

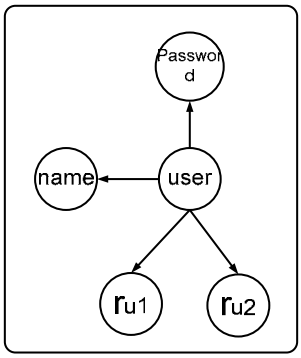

Figure 5. Fine-grained access control's rule.

The definition of the fine-grained access control for Figure 5 is:

- Components of the system

  User: The person accessing the system.

  Name: User's identification, which is unique or specific for every user.

  Password: User's security code, which verifies the identification of the user.

  Individual rule: The rule(s) that define the rights of the user in the system.

- Each vertex represents the following:

User: us = a User

Name: n = name for user

Password: p = password for user

An individual rule k: IR_k

The set of individual rules is: iru = $\{IR_1, IR_2, \ldots, IR_m\}$ where IR_k is the k^{th} individual rule for user and $k \leq m$

- The FGAC access control system is defined as follows

G = {VF, EF}

Where

$$VF = \{\text{us}, \text{n}, \text{p}\} \cup iru$$

$$EF = \{\{(\text{us}, \text{n}), (\text{us}, \text{p})\} \cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}$$

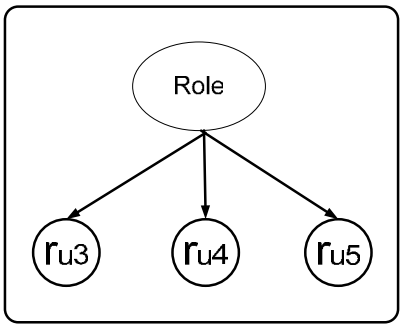

Figure 6.Role-based Access Control's Rule

The definition of the role-based access control in Figure 6 is:

- Components of the system

Role: The position of the user inside the system.

Collective or group rule: The rule(s) define the right of the role in the system.

- Each vertex represents the following:

  Role: ro = Role for the user

  A collective rule j: CR_j

  The set of collective rules is: cru = $\{CR_1, CR_2, \ldots, CR_n\}$- where CR_j is the j^{th} collective rule for user and $j \leq n$

- The RBAC access control system is defined as follows

  G = {VR, ER}

  Where

$$VR = \{\text{ro}\} \cup cru$$

$$ER = \bigcup_{j=1}^{n} \{(ro, CR_j)\}$$

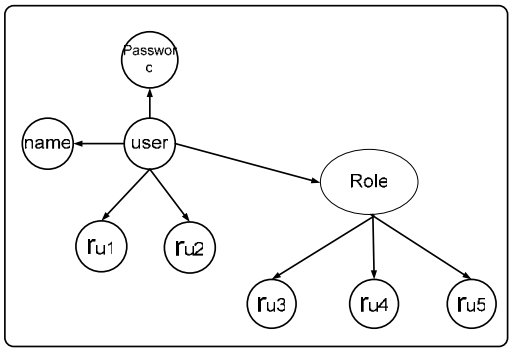

Figure 7. Fine-grained role-based access control

The rules ru1 and ru2 associated with the user define the Fine-Grained Access Control (individual rule) and the rules ru3, ru4, and ru5 associated with the role define

the Role-Based Access Control (collective rule). In the combined system, a role is associated with a user. This is shown in Figure 7. Figure 7 shows association of the user in FGAC with the role in RBAC. However, conflict and repetition between individual rule(s) and collective rule(s) can occur in this system.

A conflict happens when one or more rules contradict other rules. For example, if the individual rule defines the storage size for user 'A' as 2 Gigabyte and the collective rule defines the storage size for user 'A' as 4 Gigabyte, then a conflict occurs. Replication takes place when the existing rule is restated by another rule. For example, if the individual rule defines the storage size for user 'A' as 2 Gigabyte, and the collective rule defines the same storage size for user 'A', we have replication.

Conflict between an individual rule in FGAC and a collective rule in RBAC can arise in the combined system. For example, assume there are three data called A, B, and C. If the collective rule in the user's role permits the user to access data A and data B and forbids access to data C, but the individual rule permits the user to access data C, then a conflict occurs. This kind of problem can be solved by an FGAC priority over RBAC or an RBAC priority over FGAC setting in the system. FGAC priority over RBAC means that, when the conflict happens, the individual rule is chosen rather than the collective rule. RBAC priority over FGAC means that the collective rule is chosen rather than the individual rule when conflict occurs.

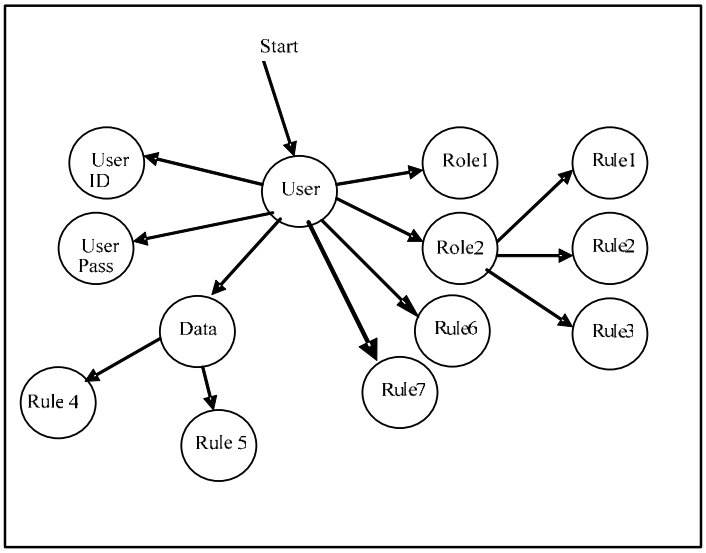

Figure 8. Graph of the Combined System

Figure 8 shows the structure of the combined system. Users enter their ID and password. In the fine-grained component of the system, there are rules associated with each individual user. These rules define how each user may access the data. Rules 6 and 7 in Figure 8. In the role-based component of the system, there are roles associated with each user. A role will have a number of associated rules. Each rule linked to a role defines the access rights associated with that role. For example, see rules 1, 2 and 3.

In order to ensure the integrity of the system, rules are associated with the data. If a user wishes to access data, any rules associated with that user and rules associated with the roles of the user must match the rules associated with the data. For example, if user wishes to access data, any of its associated rules, which include individual rules (rules 6 and 7) and collective rules that belong to the user's role (role 2 and the collective rules 1, 2, and 3) must match rule 4 and rule 5 associated with the data.

The proposed system has a number of advantages.

- One advantage of the proposed system is the reduced number of rules required. All the rules that apply to an individual user do not have to be stored. Typically, this is because there is considerable overlap in the rules associated with an individual user and a role. User-specific rules (password for example) must be stored for each individual; whereas, access rights are stored as rules associated with roles. The data does not need to maintain all of the individual user rules as in fine-grained access control. The data only needs to maintain the rules that are associated with the role of the user who is allowed to access the data. When a user requests access to data, the rules associated with the user's role are matched to the data rules specifying the role's access rights. For example, rules 4 and 5 associated with the data define that users with roles 1 and 2 are allowed to access it. Similarly rules 1, 2, and 3 associated with role 2 define the access rights of that role.
- A typical role-based system does not allow customization. However, in the proposed system, the combination of roles and individual rules provide the advantages of both systems.
- Furthermore, the proposed combined system maintains the identity of the user, providing for a more secure system since the user identity can be traced in the event of a security breach.

There are two ways to resolve the rule conflicts and rule repetition problem between FGAC and RBAC:

- Fine-grained access system (FGAC) priority over role-based access system (RBAC).

In this priority, the collective rule(s) will be removed if conflict or repetition occurs. Figure 9 shows a graph of the combined system when this priority is applied to the system shown in figure 7. Here ru1 and ru2 (individual rules) conflict or repeat with ru3 and ru4 (collective rules).

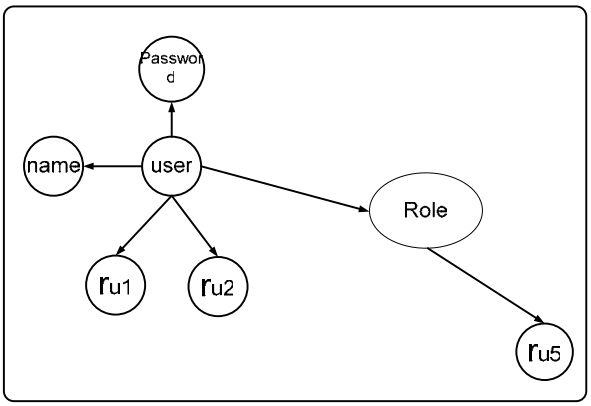

Figure 9. Fine-grained role-based access control with fine-grained priority over role-based.

- Role-based access system with priority over fine-grained access system.

  In this priority, the individual rule(s) will be removed if the individual rule(s) conflict with or duplicate collective rule(s). Figure 10 shows a graph of the combined system when this priority is applied to the system shown in Figure 7. Here ru2 (individual rule) conflicts or repeats with ru3 (collective rule):

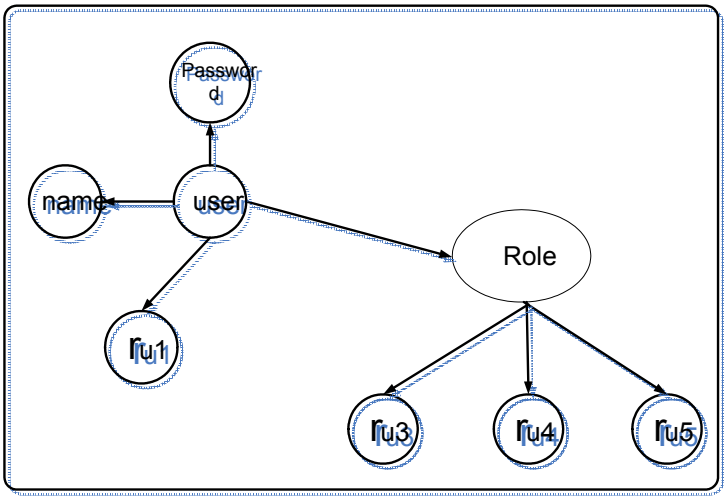

Figure 10. Fine-grained role-based access control with role-based priority over fine-grained.

There are four categories of the combined FGAC and RBAC system:

- One user with single role
- One user with many roles
- Many users with single role
- Many users with many roles.

5.1 A combined system for one user with one role.

The simplest format is the combined system for one user with one role. In this format, the system only considers a single user who has one role in the system. Here, the combined system can be categorized as RBAC priority over FGAC and FGAC priority over RBAC. Please refer to the algorithm in section 5.1.1 and 5.1.2 for the one user with one role FGRBAC system.

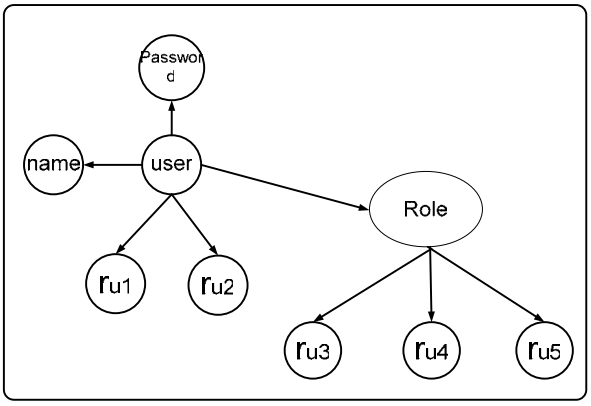

Figure 11. Combined Access Control – FGRBAC for One User to One Role

The definition of the fine-grained role-based access control for one user with single role is:

- Components of the system

  User: The person who is accessing the system.

  Name: User's identification which is unique or specific for every user.

  Password: User's security code which verifies the identification of the user

  Role: The position of the user inside the system.

  Individual rules: The rules that define the individual rights of the user.

  Collective rules: The rules that define the rights of the role to which a user belongs.

  Combined rule: The rule(s) that define the rights of the user in the system (this includes both individual and collective rules).

- Each vertex represents the following:

  User: us represents a User

  Name: n represents name for user

  Password: p represents password for user

  Role: ro represents Role for user

An individual rule k: IR_k

The set of individual rules is: iru = $\{IR_1, IR_2, \ldots, IR_m\}$ where IR_k is the k^{th} individual rule for user and $k \leq m$

A collective rule j: CR_j

The set of collective rules is: cru = $\{CR_1, CR_2, \ldots, CR_n\}$- where CR_j is the j^{th} collective rule for user and $j \leq n$

- The combined access control graph for one user with single role is defined as follows

  G = {V, E}

  Where

  $$V = \{\text{us}, \text{n}, \text{p}, \text{ro}\} \cup iru \cup cru$$

  $$E = (\text{us}, \text{n}) \cup (\text{us}, \text{p}) \cup (\text{us}, ro) \cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{j=1}^{n} \{(ro, CR_j)\})$$

5.1.1 One user with one role with RBAC priority over FGAC option.

In the category of one user with one role with RBAC priority over FGAC, when the individual rules either conflict or replicate collective rules, the individual rules will be removed. For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The algorithm for the RBAC priority over FGAC for one user with one role FGRBAC system is:

for all $IR_k \in$ iru **do**
    for all $CR_j \in$ cru **do**

{
if IR_k *conflicts* or *replicate* CR_j **then**
iru = iru – {IR_k}
V = V – {IR_k}
E = E – (us,IR_k)
}

Example: The graph for one user with one role with RBAC priority over FGAC option is:

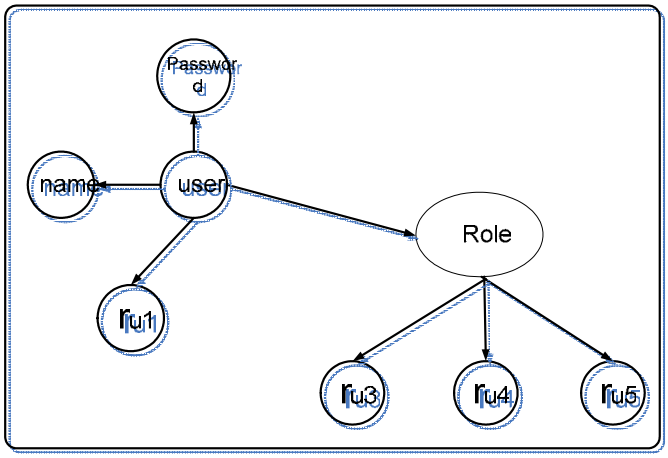

Figure 12. One user with one role with RBAC priority over FGAC.

The definition of the fine-grained role-based access control with one user to one role for role-based priority over fine-grained access control system is:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User: us represents a user

  Name: n represents name for user

  Password: p represents password for user

  Role: ro represents Role for user

  An individual rule k: IR_k

The set of individual rules is: iru = {IR_1, IR_2,…, IR_m} = {IR_1, IR_2,…, IR_f}- {CR_1, CR_2,…, CR_n}where IR_k is the k^{th} individual rule for user and $k \leq m$

A collective rule j: CR_j

The set of collective rules is: cru = {CR_1, CR_2,…, CR_n}- where CR_j is the j^{th} collective rule for user and $j \leq n$

- The one user with one role with RBAC priority over FGAC graph is defined as follows

  G = {V, E}

  Where

$$V = \{us, n, p, ro\} \cup iru \cup cru$$

$$E = (us, n) \cup (us, p) \cup (us, ro) \cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{j=1}^{n} \{(ro, CR_j)\})$$

5.1.2 A single user with a single role for FGAC priority over RBAC option.

In the one user with one role with FGAC priority over RBAC category, when the individual rules either conflict or repeat collective rules, the collective rules will be removed. For the FGAC graph, please refer to Figure 5. For the RBAC graph, please refer to Figure 6. The algorithm for the FGAC priority over RBAC in one user with one role FGRBAC system is:

for all $IR_k \in$ iru **do**
  for all $CR_j \in$ cru **do**
    {
    if IR_k *conflicts* or *replicate* CR_j **then**
      cru = cru – {CR_j}

$V = V - \{CR_j\}$

$E = E - (ro, CR_j)$

}

Example: The graph for one user with one role with FGAC priority over RBAC option is:

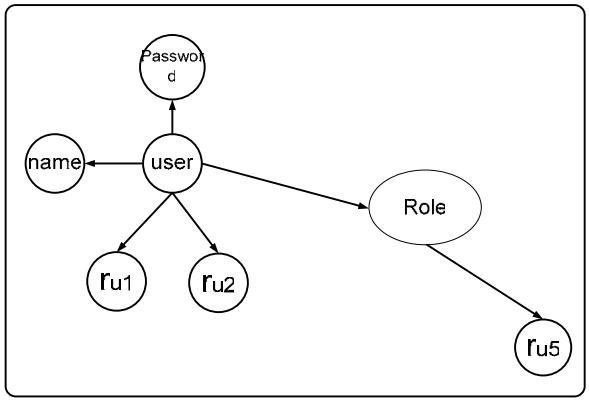

Figure 13. One user with one role with FGAC priority over RBAC.

The definition of the fine-grained role-based access control with one user to one role for fine-grained priority over role-based access control system is:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User: us represents a user

  Name: n represents name for user

  Password: p represents password for user

  Role: ro represents Role for user

  An individual rule k: IR_k

  The set of individual rules is: iru = $\{IR_1, IR_2, \ldots, IR_m\}$ where IR_k is the k^{th} individual rule for user and $k \leq m$

  A collective rule j: CR_j

The set of collective rules is: cru = $\{CR_1, CR_2,\ldots, CR_n\}$ = $\{CR_1, CR_2,\ldots, CR_g\}$ - $\{IR_1, IR_2,\ldots, IR_m\}$ where CR_j is the j^{th} collective rule for user and j $\leq$ n

- The one user with one role with FGAC priority over RBAC graph is defined as follows

  G = {V, E}

  Where

$$V = \{\text{us}, \text{n}, \text{p}, \text{ro}\} \cup iru \cup cru$$

$$E = (\text{us}, \text{n}) \cup (\text{us}, \text{p}) \cup (\text{us}, ro) \cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{j=1}^{n} \{(ro, CR_j)\})$$

5.2 A combined system for many users with single role.

This rule is used to define multiple users who have the same role or position. For example, a company which hires two or more accountants will have multiple users with the same role or category. In the many users with one role condition, the fine-grained priority over role-based access control system by eliminating the collective rule can't be realized because the collective rule (rule(s) belong to the role) is shared among more than one user. The algorithm for the RBAC priority over FGAC in one user with one role FGRBAC system is:

for all $US_t \in$ us **do**
  for all $IR_k \in iru_t$ **do**
    for all $CR_j \in$ cru **do**
      {
      if IR_k *conflicts* or *replicate* CR_j **then**
        $iru_t = iru_t - \{IR_k\}$

$V = V - \{IR_k\}$

$E = E - (US_t, IR_k)$

}

Example: The graph of many users with one role with RBAC priority over FGAC option is:

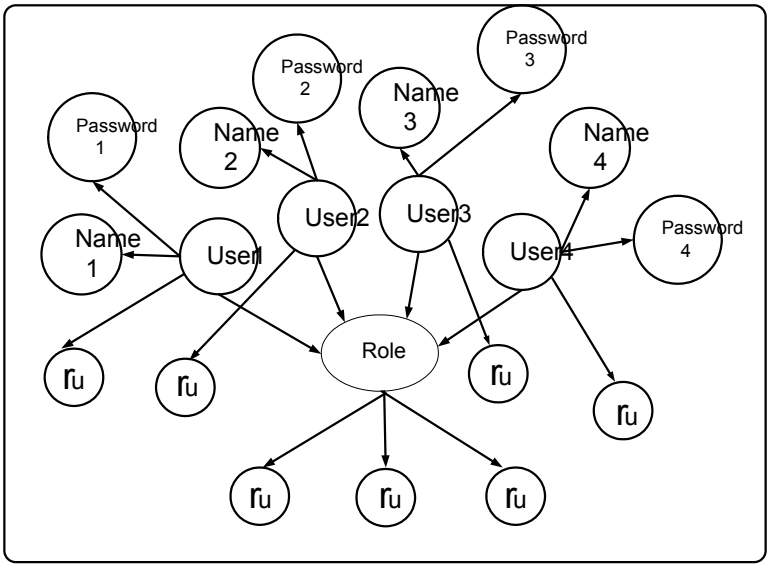

Figure 14. Graph for many users with one role for RBAC priority over FGAC.

The definition of the fine-grained role-based access control with many users to one role for RBAC priority over FGAC system is:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User t: US_t

  The set of users is: us $=\{US_1, US_2, \ldots, US_q\}$ where us_i is the i^{th} user and $i \leq q$

  Name r: N_t

The set of names is: n ={N_1, N_2,…, N_q} where n_i is the name for i^{th} user and $i \leq q$

Password s: P_s

The set of password is: p ={P_1, P_2,…, P_q} where p_i is the password for i^{th} user and $i \leq q$

Role: ro represents the single Role for all the users

An individual rule k: IR_k

The set of individual rules is: iru = {IR_1, IR_2,…, IR_m} = {IR_1, IR_2,…, IR_f}- {CR_1, CR_2,…, CR_n}where IR_k is the k^{th} individual rule for all users and $k \leq m$

A collective rule j: CR_j

The set of collective rules is: cru = {CR_1, CR_2,…, CR_n}- where CR_j is the j^{th} collective rule for i^{th} user and $j \leq n$

The set of collective rules for a user i: $cru_i \subseteq cru$

- The many users with one role with RBAC priority over FGAC graph is defined as follows

G = {V, E}

Where

$$V = us \cup n \cup p \cup \{\text{ro}\} \cup iru \cup cru$$

$$E = (\bigcup_{t=1}^{q}\{(US_t, N_t)\}) \cup (\bigcup_{t=1}^{q}\{(US_t, P_t)\}) \cup (\bigcup_{t=1}^{q}\{(US_t, ro)\})$$

$$\cup (\bigcup_{t=1}^{q} \bigcup_{k \in iru_t}\{(US_t, IR_k)\}) \cup (\bigcup_{j=1}^{n}\{(ro, CR_j)\})$$

$$, where\ iru_t \in\ iru\ and\ t\ =\ 1,2,...,q$$

5.3 Combined system for single user with many roles.

This category can be described as a user who has more than one position in the system. An example is a user who is the president and also the CEO of a company. In this category, the combined system can be categorized as RBAC priority over FGAC and FGAC priority over RBAC. Refer to the algorithm in section 5.3.1 and 5.3.2 for the one user with many roles FGRBAC system.

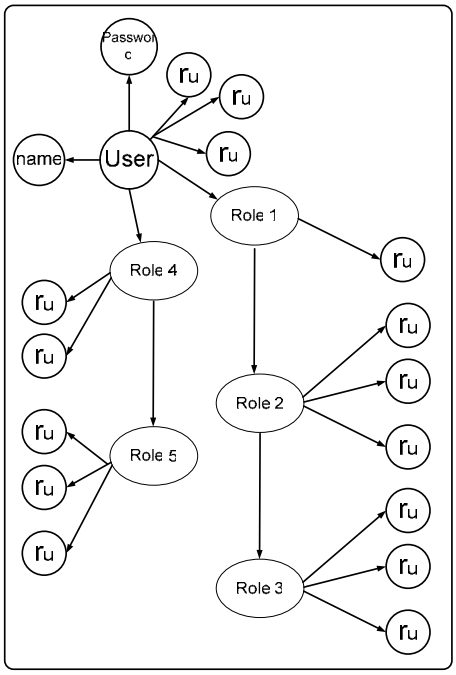

Figure 15. Combined access control – FGRBAC for one user to many roles

The definition of the fine-grained role-based access control with single user to many roles for RBAC priority over FGAC system in the hierarchy role:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User: us represents a user

  Name: n represents name for user

  Password: p represents password for user

  Role o: RO_o

  The set of roles is: ro ={RO_1, RO_2,…, RO_p} where ro_o is the o^{th} role for user and $o \leq q$

  A collective rule j: CR_j

The set of collective rules is: cru = {CR_1, CR_2,…, CR_n}- where CR_j is the j^{th} collective rule for the user and $j \leq n$

An individual rule k: IR_k

The set of individual rules is: iru = {IR_1, IR_2,…, IR_m}- where IR_k is the k^{th} individual rule for the user and $k \leq m$

- The one user with one role with RBAC priority over FGAC graph is defined as follows:

  G = {V, E}

  Where

$$V = \{\text{us}, \text{n}, \text{p}\} \cup ro \cup iru \cup cru$$

$$E = \{(\text{us}, \text{n}), (\text{us}, \text{p})\} \cup (\bigcup_{o=1}^{p} \{(us, RO_o)\}) \cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$, where\ cru_o \in cru\ and\ o = 1,2,...,p$$

5.3.1 One user with many roles with RBAC priority over FGAC option.

In one user with many roles with RBAC priority over FGAC option category, when the individual rules either conflict or repeat over with collective rules, the individual rules will be removed. In this system, the role in the company can be categorized either as a hierarchy role or a non-hierarchy role [Wainer, 2005]. In a

Hierarchical role, a user is directed to one role and other roles are inherited from that role. Inheritance in this paper means all the rules belong to the role, which is connected to the upper most roles (the role connected to the user), are inherited partially or totally from that upper most role. For example, if the user is a CEO of a company and the user can also have a role as a manager or accountant of the company then these roles (manager and accountant) are inherited from the CEO role. A non-hierarchy role is the condition when a user is directed to many roles without inheritance. For example, if the user is a CEO of a company and the user is also a manager of the company, where this role, manager, is not inherited from the CEO. The algorithm for the RBAC priority over FGAC in one user with many roles FGRBAC system for either hierarchy or non-hierarchy role is:

for all $IR_k \in$ iru **do**
  for all $RO_o \in$ ro **do**
    for all $CR_j \in$ cru **do**
      {
      if IR_k *conflicts* or *replicate* CR_j **then**
        iru = iru – $\{IR_k\}$
        V = V – $\{IR_k\}$
        E = E – (us, IR_k)
      }

5.3.1.1 Hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for one user with many roles with RBAC priority over FGAC option in the hierarchy role is:

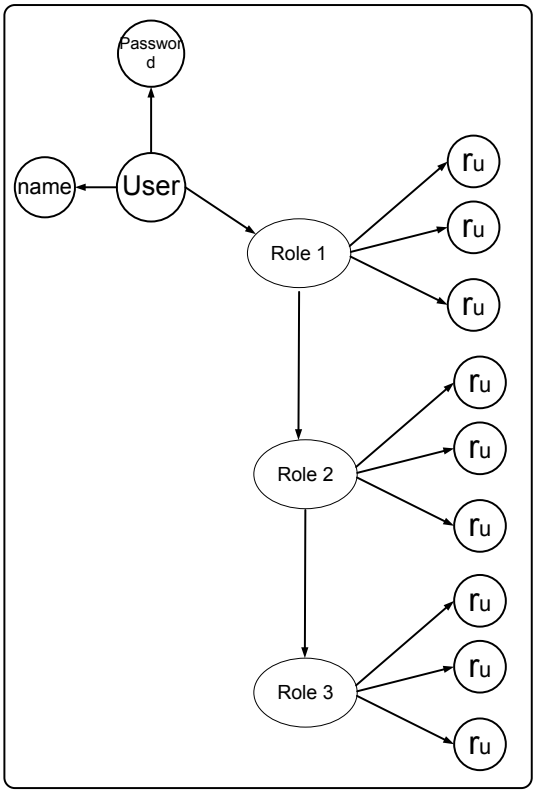

Figure 16. One User with Many Roles with RBAC Priority over FGAC System and the Role System is Hierarchy Role

The definition of the fine-grained role-based access control with single user to many roles for RBAC priority over FGAC system in the hierarchy role:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User: us represents a user

  Name: n represents name for user

  Password: p represents password for user

  Role o: RO_o

  The set of roles is: ro ={RO_1, RO_2,…, RO_p} where ro_o is the o^{th} role for user and $o \leq q$

A collective rule j: CR_j

The set of collective rules is: cru = $\{CR_1, CR_2,\ldots, CR_n\}$- where CR_j is the j^{th} collective rule for the user and $j \leq n$

An individual rule k: IR_k

The set of individual rules is: iru = $\{IR_1, IR_2,\ldots, IR_m\}$= $\{IR_1, IR_2,\ldots, IR_f\}$- $\{CR_1, CR_2,\ldots, CR_n\}$ where IR_k is the k^{th} individual rule for the user and $k \leq m$

- The one user with many roles with RBAC priority over FGAC graph for hierarchy role is defined as follows:

  G = {V, E}

  Where

$$V = \{\text{us}, \text{n}, \text{p}\} \cup ro \cup iru \cup cru$$

$$E = \{(\text{us}, \text{n}), (\text{us}, \text{p}), (\text{us}, RO_1)\} \cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$, where\ cru_o \in\ cru\ and\ o\ =\ 1,2,...,p$$

5.3.1.2 Non-hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for one user with many roles with RBAC priority over FGAC option in the non-hierarchy role is:

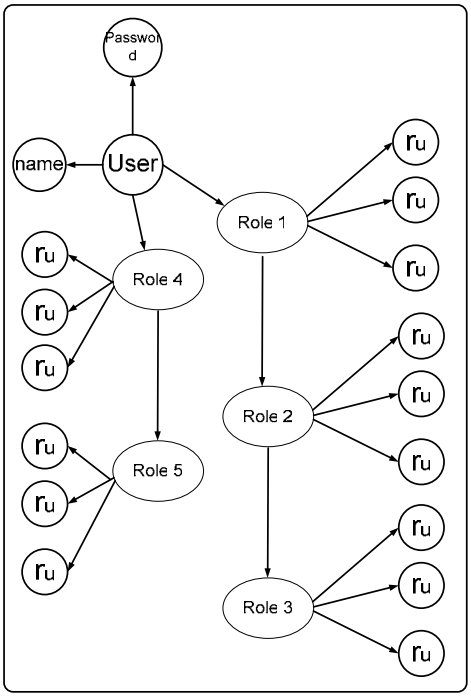

Figure 17. One User with Many Roles with RBAC Priority over FGAC System and the Role System is Non-HIERARCHICAL

The definitions of the fine-grained role-based access control with single user to many roles for RBAC priority over FGAC system in the non-hierarchy role are:

- Refer to section 5.3.1.1 for definition of components, vertices representation of the system.
- The one user with many roles with RBAC priority over FGAC graph for non-hierarchy role is defined as follows:

  G = {V, E}

  Where

$$V = \{\text{us}, \text{n}, \text{p}\} \cup ro \cup iru \cup cru$$

$$E = \{(\text{us}, \text{n}), (\text{us}, \text{p})\} \cup (\bigcup_{o=1}^{p} \{(us, RO_o)\}) \cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$, where\ cru_o \in cru\ and\ o = 1,2,...,p$$

5.3.2 One user with many roles with FGAC priority over RBAC option.

In the one user with many roles with FGAC priority over RBAC option category, when the individual rules either conflict or duplicate the collective rules, the collective rules will be removed. In this system, the role in the company can be categorized as hierarchy role and non-hierarchy role [Wainer, 2005]. The algorithm for the FGAC priority over RBAC in one user with one role FGRBAC system for both hierarchy and non-hierarchy role is:

for all $IR_k \in$ iru **do**
  for all $RO_o \in$ ro **do**
    for all $CR_j \in$ cru **do**
      {
      if IR_k *conflicts* or *replicate* CR_j **then**
        iru = iru – { CR_j }
        V = V – { CR_j }
        E = E – (RO_o, CR_j)
      }

5.3.2.1 Hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for one user with many roles with FGAC priority over RBAC option in the hierarchy role is:

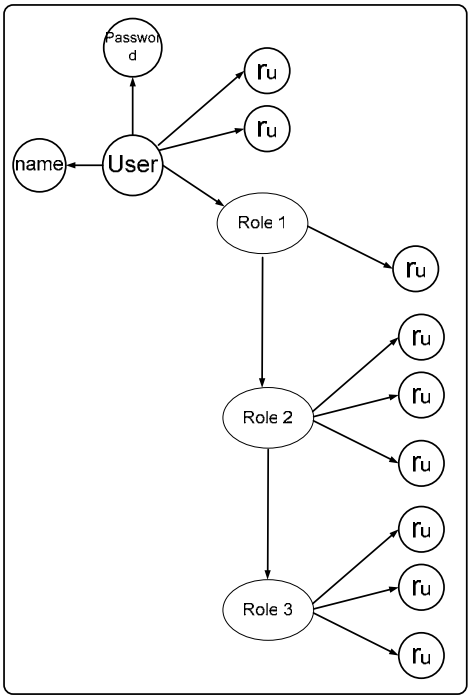

Figure 18. One User with Many Roles with FGAC priority over RBAC System and the Role System in Hierarchy Role

The definition of the fine-grained role-based access control with single user to many roles for RBAC priority over FGAC system in the hierarchy role is:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User: us represents a user

  Name: n represents name for user

  Password: p represents password for user

Role o: RO_o

The set of roles is: ro = $\{RO_1, RO_2, \ldots, RO_p\}$ where ro_o is the o^{th} role for the user and $o \leq q$

An individual rule k: IR_k

The set of individual rules is: iru = $\{IR_1, IR_2, \ldots, IR_m\}$ - where IR_k is the k^{th} individual rule for the user and $k \leq m$

A collective rule j: CR_j

The set of collective rules is: cru = $\{CR_1, CR_2, \ldots, CR_n\}$ = $\{CR_1, CR_2, \ldots, CR_g\}$ - $\{IR_1, IR_2, \ldots, IR_m\}$ - where CR_j is the j^{th} collective rule for the user and $j \leq n$

- The one user with many roles with FGAC priority over RBAC graph for hierarchy role is defined as follows:

  G = {V, E}

  Where

$$V = \{\text{us}, \text{n}, \text{p}, \text{ro}\} \cup ro \cup iru \cup cru$$

$$E = (\text{us}, \text{n}) \cup (\text{us}, \text{p}) \cup (\text{us}, RO_1) \cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$, where\ cru_o \in cru\ and\ o = 1,2,...,p$$

5.3.2.2 Non-hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for one user with many roles with FGAC priority over RBAC option in non-hierarchy role is:

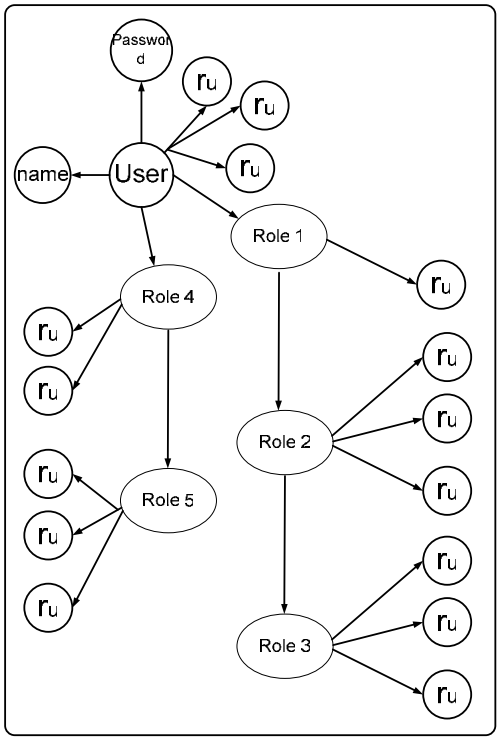

Figure 19. One User with Many Roles with FGAC Priority over RBAC System and the Role System is Non-hierarchy Role

The definition of the fine-grained role-based access control with single user to many roles for FGAC priority over RBAC system in the non-hierarchy role is:

- Please refer to section 5.3.2.1 for definition of components, vertices representation of the system.
- The one user with many roles with FGAC priority over RBAC graph for non-hierarchy role is defined as follows:

  G = {V, E}

Where

$$V = \{\text{us}, \text{n}, \text{p}\} \cup ro \cup iru \cup cru$$

$$E = (\text{us}, \text{n}) \cup (\text{us}, \text{p}) \cup (\bigcup_{o=1}^{p} \{(us, RO_o)\}) \cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{k=1}^{m} \{(us, IR_k)\}) \cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$, where\ cru_o \in cru\ and\ o = 1,2,...,p$$

5.4 A combined system for many users with many roles.

This category is the most complex situation. This category can be described as the combination of all the rules above. In this many users with many roles, the fine-grained priority over role-based access control by eliminating the collective rule is impossible because the collective rule (rule(s) belonging to the role) is shared by more than one user. Please refer to the algorithm in section 5.4.1 for the one user with many roles FGRBAC system.

5.4.1 Many users with many roles with RBAC priority over FGAC.

In the many users with many roles with RBAC priority over FGAC option, when the individual rules either conflict or are identical to collective rules, the individual rules will be removed. In this system, the roles in the company can be categorized as hierarchy and non-hierarchy roles [Wainer, 2005]. The algorithm for the RBAC priority over FGAC in a one user with one role FGRBAC system for both hierarchy and non-hierarchy role is:

for all $US_t \in$ us **do**

for all $IR_k \in iru_t$ **do**
    for all $RO_o \in ro_t$ **do**
        for all $CR_j \in cru$ **do**
            {
            if IR_k *conflicts* or *replicate* CR_j **then**
                $iru_t = iru_t - \{ IR_k \}$
                $V = V - \{ IR_k \}$
                $E = E - (US_t, IR_k)$
            }

5.4.1 The hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for one user with many roles with RBAC priority over FGAC option in hierarchy role is:

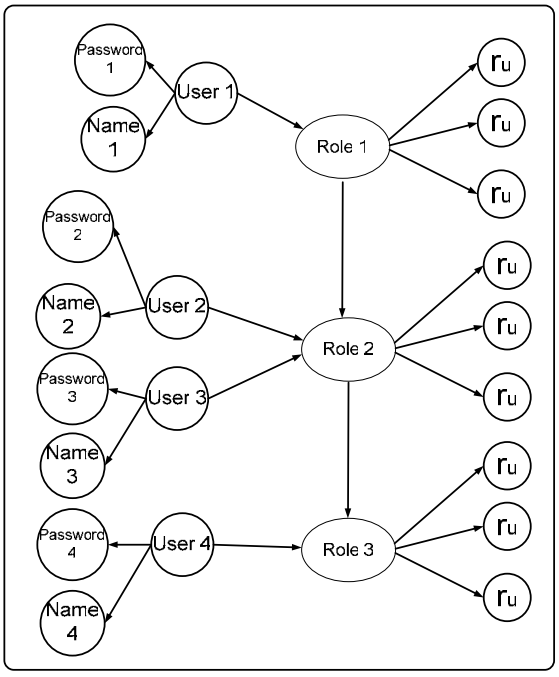

Figure 20. Many Users with Many Roles for RBAC Priority over FGAC System in Hierarchy Role

The definition of the fine-grained role-based access control with many users to many roles for RBAC priority over FGAC system in the hierarchy role is:

- Refer to section 5.1 for definition of components.
- Each vertex represents the following:

  User t: US_t

  The set of users is: us $=\{US_1, US_2,\ldots, US_q\}$ where us_i is the i^{th} user and $i \leq q$

  Name r: N_t

  The set of names is: n $=\{N_1, N_2,\ldots, N_q\}$ where n_i is the name for i^{th} user and $i \leq q$

  Password s: P_s

  The set of password is: p $=\{P_1, P_2,\ldots, P_q\}$ where p_i is the password for i^{th} user and $i \leq q$

  Role o: RO_o

  The set of roles is: ro $=\{RO_1, RO_2,\ldots, RO_p\}$ where ro_o is the o^{th} role for user i and $o \leq p$

  An individual rule k: IR_k

  The set of individual rules is: iru $= \{IR_1, IR_2,\ldots, IR_m\} = \{IR_1, IR_2,\ldots, IR_f\} - \{CR_1, CR_2,\ldots, CR_n\}$- where IR_k is the k^{th} individual rule for us i^{th} user and $k \leq m$

  The set of individual rules for a user i: $iru_i \subseteq$ iru

  A collective rule j: CR_j

The set of collective rules is: cru = {CR_1, CR_2,…, CR_n}- where CR_j is the j^{th} collective rule for i^{th} user and $j \leq n$

The set of collective rules for i^{th} user: $cru_i \subseteq cru$

- The many users with many roles with RBAC priority over FGAC graph for hierarchy role is defined as follows:

  G = {V, E}

  Where

  $V = us \cup n \cup p \cup ro \cup iru \cup cru$

$$E = (\bigcup_{t=1}^{q} \{(US_t, N_t)\}) \cup (\bigcup_{t=1}^{q} \{(US_t, P_t)\})$$

$$\cup (\bigcup_{t=1}^{q} \bigcup_{o=1}^{p} \{(US_{o_t}, RO_o)\})$$

$$\cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$\cup (\bigcup_{t=1}^{q} \bigcup_{k \in iru_t} \{(US_t, IR_k)\})$$

$, where\ cru_o \in cru$

$, iru_t \in iru$

$, t = 1,2,...,q$

$, and\ o = 1,2,...,p$

5.4.2 Non-hierarchy Role.

For the FGAC graph, refer to Figure 5. For the RBAC graph, refer to Figure 6. The graph for many users with many roles with RBAC priority over FGAC option in non-hierarchy role is:

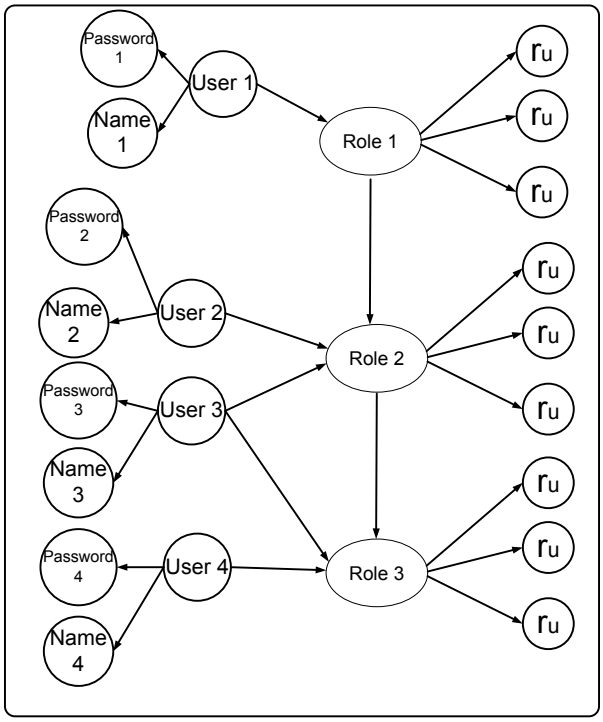

Figure 21. Many Users with Many Roles for RBAC Priority over FGAC System in Non-hierarchy Role

The definition of the fine-grained role-based access control with many users to many roles for RBAC priority over FGAC system in the non-hierarchy role is:

- Please refer to section 5.4.1 for the definition of components, vertices, and representation of the system.

- The many users with many roles with RBAC priority over FGAC graph for non-hierarchy role is defined as follows:

G = {V, E}

Where

$$V = us \cup n \cup p \cup ro \cup iru \cup cru$$

$$E = (\bigcup_{t=1}^{q} \{(US_t, n_t)\})$$

$$\cup (\bigcup_{t=1}^{q} \{(US_t, P_t)\})$$

$$\cup (\bigcup_{t=1}^{q} \bigcup_{o \in ro_t} \{(US_t, RO_o)\})$$

$$\cup (\bigcup_{o=1}^{p} \{(RO_o, RO_{o+1})\})$$

$$\cup (\bigcup_{o=1}^{p} \bigcup_{j \in cru_o} \{(RO_o, CR_j)\})$$

$$\cup (\bigcup_{t=1}^{q} \bigcup_{k \in iru_t} \{(US_t, IR_k)\})$$

$$, where\ cru_o \in cru$$

$$, iru_t \in iru$$

$$, ro_t \in ro$$

$$, t = 1,2,...,q$$

$$, and\ o = 1,2,...,p$$

Varieties of the combined access control system are shown in this chapter. The varieties of one user with one role, one user with many roles, many users with one role, and many users with many roles are explained. In addition to the varieties of users against roles, FGAC priority over RBAC and RBAC priority over FGAC are also

explained in this chapter. In the next chapter, the result sample of the simulation is shown.

Chapter 6

Simulation

Simulation results in this chapter show the differences of information kept by the user, data, and roles in the FGAC, RBAC, and FGRBAC systems. This information shows the FGRBAC system is more secure than the RBAC system and simpler than the FGAC system.

A simulation program was implemented using the Java programming language. The structures of the program used are fixed variables, and simple "if….else" statements. The program requires input from the user for systems such as FGAC priority over RBAC or RBAC priority over FGAC, user ID and user password. After the user enters these three items of information, the program lets the user know what information is stored in the user, data, and role (if available, i.e., RBAC and FGRBAC systems) variables. From this information, the simplicity and the security structure can be determined from among FGAC, RBAC and FGRBAC system. There are three users, two roles, and two files. The program generates and shows the user's individual rules, information carried by data, and the role's rules from the information entered by the user.

The information displayed by the simulation is:

Table 1. Comparison of Information Stored in FGAC, RBAC, and FGRBAC

	Information stored in user	Information stored in data	Information stored in role
FGAC	User ID User password Access rights for specific file	User ID of the user allowed to access the data Password of the user allowed to access the data Access rights of the specific user	
RBAC	User ID User password Role of the user	Role that is allowed to access the data Access right of each role	Password of the role
FGRBAC	User ID User password Role of the user Access rights for specific user (Individual Rule)	Role that is allowed to access the data Access right of each role	Password of the role User ID of the accessing user Access right of each role (Collective Rule)

The results in Table 1 show the differences of the information stored in user, data, and role between FGAC, RBAC, and FGRBAC. The FGRBAC system gives more security than the RBAC system because it provides information about the users who access the specific files. The information of the user can be retrieved from the role information as can be seen in Table 1. In the RBAC system, the role cannot store information about the user accessing the data. In addition, the FGRBAC system requires less information stored in the user and data than the FGAC system. When a user accesses the data in the system, the data needs to check the ID, password, and the access rights of the user. Table 1 show the FGRBAC system only requires the data to check the role of the user and the access rights of the role instead of the ID, password, and access rights of the user as in the FGAC system. In addition to the data stored in the data inside the FGRBAC system, the user does not need to store all the access rights since all the access rights have been stored in the role. Refer to Table 2 for example of FGAC system, Table 3 for example of RBAC system, and Table 4 for example of the combined FGRBAC system.

Table 2. Example of Information Stored in User, and File for FGAC System

Example of the information stored in FGAC system			
	User A	User B	User C
Information stored in user	ID : A Password : A Access Right for: - File 'A' : Read, Write - File 'B' : Read	ID : B Password : B Access Right for: - File 'A' : Read - File 'B' : Read, Write	ID : C Password : C Access Right for: - File 'A' : Read - File 'B' : Not Available
Information stored in File 'A'	ID : A Password : A Access Right : Read, Write	ID : B Password : B Access Right : Read	ID : C Password : C Access Right : Read
Information stored in File 'B'	ID : A Password : A Access Right : Read	ID : B Password : B Access Right : Read, Write	ID : C Password : C Access Right : None

Table 3. Example of Information Stored in User, Role, and File for RBAC System

Example of the information stored in RBAC system			
	User A	User B	User C
Information stored in user	ID : A Password : A Role : Student	ID : B Password : B Role : - Student - Teaching Assistant	ID : C Password : C Role : Teaching Assistant
Information stored in role : Student	Password: Stu	Password: Stu	Password: Stu
Information stored in Teaching Assistant	Password: TA	Password: TA	Password: TA
Information stored in File 'A'	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write
Information stored in File 'B'	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write

Table 4. Example of Information Stored in User, Role, and File for FGRBAC System

Example of the information stored in FGRBAC system			
	User A	User B	User C
Information stored in user	ID : A Password : A Role : Student Email Storage Size: '2 GB'	ID : B Password : B Role : - Student - Teaching Assistant Email Storage Size: '2 GB'	ID : C Password : C Role : Teaching Assistant Email Storage Size: '2 GB'
Information stored in role : Student	Password: Stu Accessing user : A Email Storage Size: '2 GB'	Password: Stu Accessing user : B Email Storage Size: '2 GB'	Password: Stu Accessing user : C Email Storage Size: '2 GB'
Information stored in Teaching Assistant	Password: TA Email Storage Size: '4 GB'	Password: TA Email Storage Size: '4 GB'	Password: TA Email Storage Size: '4 GB'
Information stored in File 'A'	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - Read Access Right for role "Teaching Assistant" - Read - Write
Information stored in File 'B'	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write	Access Right for role "Student" - None Access Right for role "Teaching Assistant" - Read - Write

As more information is stored in the data and user areas, the more complex a system becomes. This result shows that the FGRBAC is much better than the FGAC and RBAC system because it is more secure than the RBAC system and less complex than

the FGAC system. Table 5 show the reduced information stored in FGRBAC system compared to FGAC system.

Table 5. Number of Information Stored in FGAC compared to FGRBAC

Number of data stored in the System				
		Information load for		
		FGAC system	RBAC system	FGRBAC system
Information load for	2 Users	13	13	13
	2 Roles	-	2	6
	2 Files	19	8	15
	Total	32	23	34
Number of comparisons		30	24	30
Information load for	10 Users	60	40	40
	2 Roles	-	2	14
	2 Files	80	8	15
	Total	140	50	69
Number of comparisons		100	24	100
Information load for	30 Users	180	120	120
	2 Roles	-	2	33
	2 Files	240	8	120
	Total	420	130	273
Number of comparisons		300	24	300

Note: Number of comparison is comparisons between information stored in the data and collective rules. The more comparisons, the more secure a system.

Chapter 7

Algorithm and Tools to Support the Combined System

In chapter 5, we proposed different combinations of the FGAC and RBAC system in a graphical format. The basic foundation of the combined system is based on the individual rules of the FGAC system and the collective rules of the RBAC system.

7.1 Algorithm of the combined FGAC and RBAC system.

The data must check the user's access right before it allows the user to access a database in the system. The data will check from the individual rules of the user before it goes to the collective rules of the user's role. The algorithm is used to support the combined system by simplifying the idea in pseudocode. The algorithm representation of the system is:

FOR all rules of the user (individual rules)

IF rule of the user match the data access right

THEN user is allowed to access the data

ELSE

FOR all user's role

FOR all rules of the user's role (collective rules)

IF rule of the user's role match the data access right

THEN user is allowed to access the data

ELSE user is rejected to access the data

7.2 Automaton for the Combined FGAC and RBAC System

Automaton is used in our proposal in order to express the paths taken by the processes to match the rules (individual and collective) from the user with the information kept by the data in the FGRBAC system. Each node in the automaton represents either the individual user itself or the role of the user. Contained inside the node are the rules to decide whether the access is accepted or rejected when compared to the information inside the data. The acceptance is represented by 'a' and rejection is represented by 'r'. These 'a' and 'r' are used in both automaton and grammar. The automaton used is Deterministic Finite Automaton (DFA). DFA is used to define the combined system as an abstract model. The automaton facilitates implementation of the combined system in a programming language. If nondeterministic finite automaton (NFA) is selected instead of DFA, it must be translated into DFA before it can be implemented in the computer languages for easier translation, which involves more steps and hence is less efficient. The DFA graph is:

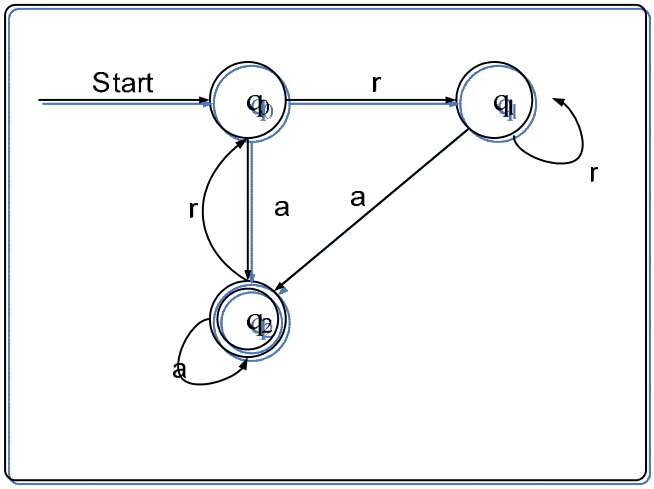

Figure 22. DFA for Combined RBAC and FGAC System with k =4

The definition of Deterministic Finite Automaton (DFA) representations when k = 4 => (*{q0, q1, q2},* {a, r}, q0, *{q2}*)

q = user or user's role where the individual rule(s) or collective rule(s) belong to.

a = accepted

r = rejected

q0 = the starting state, it represents the user (where individual rule(s) belong to).

q1, q2 = user's role (where collective rule(s) belong to)

q2 = the condition or role when the request of a certain data is accepted (can be defined as accepted state)

→ = direction of the action taken.

For example, if the permission of the access control process is: "rrrrra", then it will start from q0 and the process is rejected ('r') and it turns to q1. From q1, it goes to the second rejection ('r') and it goes to q2. From q2, it has another rejection ('r') and will go back to q2. The rejection after q2 will cause a recursive process in q2 until the access permission

is accepted (‘a’), which is the last process in “rrrrra”. With a result of ‘a’ from q2 goes to q3, which is an accepted state. This process means the permission is granted.

7.3 Grammar for the combined FGAC and RBAC system.

The grammar in this paper shows the languages accepted by the automaton defined in section 7.2. If the language is supported by this grammar, then the language can be implemented in the FGRBAC system because it also is supported by the DFA defined in section 7.2. The example below is representing the accepting ‘a’ or rejection ‘r’ of the access permission in each node in the DFA. In order to introduce the combined system mathematically in a formal way, grammar is used. The grammar used in this research is a regular grammar. The language recognized from the DFA in Figure 19 is:

$\{a, r\}^* \{a\}$

Suppose we take a string, say x = rraarrara, the trace of its processing by the finite automaton is:

Substring processed so far	State
Λ	q0
r	q1
rr	q1
rra	q2
rraa	q2
rraar	q0
rraarr	q1
rraarra	q2
rraarrar	q0
rraarrara	q2

Table 5. Substring Process with DFA in Figure 19 for rraarrara

If the line of Table One is listed consecutively with the separation of =>, then we obtain the derivation of:

q0 => r*q1* => rr*q1* => rra*q2* => rraa*q2* => rraar*q0* => rraarr*q1* => rraara*q1* => rraarar*q0* => rraararaq2

The production of the form used in this system is:

1. P → a
2. P → aP
3. P → rP

With the application of the production above, the last step in the derivation becomes:

rraarar*q0* => rraarara

The inferred string based on the production used in the system is:

String Inferred	For the language of	Production Used	String Used
(i)	r	rP	Λ
(ii)	rr	rP	r
(iii)	rra	aP	rr
(iv)	rraa	aP	rra
(v)	rraar	rP	rraa
(vi)	rraarr	rP	rraar
(vii)	rraarra	aP	rraarr
(viii)	rraarrar	rP	rraarra
(ix)	rraarrara	a	rraarrar

Table 6. Inferred string for the string or rraarrara

7.4 A proof for the Combined FGAC and RBAC System

In order for the process to be accepted by the FGRBAC system, the following proposition must be satisfied:

Proposition: *The access rights information of the data has to match the rule(s) of the user (the rules belong to the user or individual rules) or the collective rule(s) (the rules belong to the user's role) for a user to access the data..*

The DFA in section 7.2 has to be true in order for the above proposition to be true. Mathematical induction is used to prove the above proposition based on the DFA in section 7.2 and the grammar in section 7.3.

The mathematical induction representation is:

$$\bigcup_{i=1}^{k} \delta\{s_i, (a, r)\}$$

k = number of user's role + 1, where 1 represents the user itself.

i = user (represented by *q0* in DFA section 7.2) or the i^{th} role of the user (represented by q_1, q_2 in DFA section 7.2)

δ = the transition state from current state (S_i) with the result of the current state (*a, r*)

S_i = the current state of the system, where S_1 is the user and S_2, S_3 are the user's role.

a = 'accepted' request

r = 'rejected' request

Basis: δ' (*s*, *Є*) = *s*. That is, if we are in state 's' and read no inputs, then we are still in state *s*. In another word, if the state 's' is not an accepted state and the information inside the data does not match any of the rules (individual or collective rules), then the request will be denied or rejected.

Induction: If the result of the match between the rules inside the data and the rules of the user (individual rules and collective rules) is simplified as '*r*' as rejected and '*a*' as accepted, the results can be formed as a string. Suppose *w* is a string that results from the match between data's rules and user's rules, *w* can be categorized as the form of *xy*; that is, *y* is the last symbol or character of *w*, and *x* is the string consisting of all but the last

symbol. For example, $w = rrrra$ is broken into $x = rrrr$ and $y = a$. Then $\delta'(s, w) = \delta(\delta'(s, x), a)$.

In order to compute $\delta(s, w)$, $\delta'(s', x)$ must be computed. The state of the automaton after processing $\delta'(s', x)$ is in all but the last symbol of w. Suppose this state is p; that is $\delta'(s', x) = p$. Then $\delta(s, w)$ is what we get by making a transition from state p on input r, the last symbol of w. That is, $\delta(s, w) = \delta'(p, r)$. This induction shows that the result is accepted because 'r', the last input, is an accepted match.

The algorithm in section 7.1 shows the basic programming logic of the system in the FGRBAC system. The expression of the path taken in the system is represented by DFA in section 7.2. The language formed from the result of the path taken in DFA is supported by grammar in section 7.3. The rule to define the system access process is supported by mathematical induction.

7.5 Implementation of the FGRBAC System

The FGRBAC system can be implemented in Java. Graphs in the FGRBAC system can be represented in a tree structure in the implementation of the system. Insertion or deletion of users, roles, and rules inside the FGRBAC system can be realized by categorizing the users, roles, and rules using XML transformation language. XML is extensible Mark-up Language, a specification developed by the World Wide Web Consortium**.** The reason for using the XML transformation language is the flexibility of XML. If the users, roles, and rules are categorized in XML transformation language, it can be used by any languages in addition to JAVA for the delegation and revocation process of the system. For example, the XML implementation from figure 7 is:

```
<user1>
        <name> John </name>
        <password> john </password>
        <rule1> .... </rule1>
        <rule2> .... </rule2>
        <role>
                <rule3> .... </rule3>
                <rule4> .... </rule4>
                <rule5> .... </rule5>
        </role>
</user1>
```

The implementation of the XML in the system above shows its flexibility. When the rules or role needs to be delegated or revoked, it can be easily done by recognition of the tag name. The delegation and revocation process can be accomplished with any languages if the structure of the system is implemented in XML format as in the previous example.

Chapter 8

Conclusion

Although the development of access control systems has improved, problems can be found in these systems such as the complexity problem in FGAC and lack of desired security in RBAC. Combining the FGAC and RBAC system can solve the complexity and insecurity problems.

Prior to this research, existing access control systems were based on either FGAC or RBAC systems. Any improvement that had been done added more layers of roles in the RBAC system. This kind of improvement is not secure enough due to the lack of user identity in the system. In addition to the improvement in the RBAC system, the complexity of information needs to be carried by the data in the FGAC system. Introduction of the combined system is a big step in improving the access control system. The combined system, FGRBAC, is implemented by injecting the user information into the RBAC system.

In the combined system, there are two problems: repetition and conflict between individual rules and collective rules. The solution to these problems is to eliminate the duplicated or conflicting rules. The way to eliminate these duplications and conflict is to choose either the FGAC priority over RBAC method or the RBAC priority over FGAC method. These two methods can be implemented in the system in the following format:

one user one role, one user many roles, many users one role, or many users with many roles. Chapter 5 includes a detailed explanation of the solution.

One of the advantages of this system is the reduced information load required by the user and the data in the system compared to the pure FGAC system. With this system, not all the rules must be store for the user. Only part of the user's individual rules and the roles are stored. Data within the system does not have to keep all users' information as in FGAC system. The data is required to keep the role of the user who is allowed to access the data. When the user requests access to the data, the user's roles are matched to the data's rule to permit or deny access to the data.

The other advantage of this system relates to the security issue. The FGRBAC system provides a more secure system compared to the RBAC system. In the FGRBAC system, the user ID is kept inside the data to backtrack when there is a security breach. In the pure RBAC system, this method is not provided. When a security breach occurs in an RBAC system, it is impossible to backtrack a use since the data does not keep the user's identification. Refer to Table 1 in chapter 6 for a better understanding of the differences in FGAC, RBAC, and FGRBAC system.

We have used an automaton, a grammar, and a proof to validate the FGRBAC system. The automaton used in this research is a deterministic finite automaton (DFA). The automaton is used to express the paths taken by the system to match the rules (individual rules and collective rules) of the user with the information kept by the data in the FGRBAC system. A DFA is selected to save the running time of the system implementation. If nondeterministic finite automaton (NFA) is selected instead of DFA, translation from NFA to DFA is required before the system implementation.

The grammar used in this research is a regular grammar. A regular grammar is selected to show the languages that are accepted by the automaton. If the language is supported by the grammar in chapter 7, it is certain that the language can be implemented in the FGRBAC system. The mathematical induction in chapter 7 is used to prove that the FGRBAC theorem has to be true for the system implementation based on the automaton and grammar in chapter 7.

The research done for this thesis combines the FGAC and RBAC systems. Further research can be done by implementing the delegation and revocation of the users and their roles in the FGRBAC system. Delegation and revocation of the FGRBAC system will show how users and their roles are added or removed from the system. The system can be used by any computer system with the implementation of delegation and revocation in the FGRBAC system. In addition to delegation and revocation, this system can also be implemented in XML transformation language. The implementation of the FGRBAC system can show the flexibility of this system to be implemented in various machines.

References

[Ahn, 2000] Ahn G.J., Sandhu R., Kang M., "Injecting RBAC to secure a web-based workflow system.", Fifth ACM Workshop on Role-Based Access Control, Vol. 5, No.1, 2000.

[Andrei , 2005] Andrei S.L., "Information retrieval in current research information system", http://arxiv.org/ftp/cs/papers/0110/0110026.pdf [April 5, 2005].

[Bassam , 2005]Bassam A., "Agent technology in electronic commerce and information retrieval on the Internet", http://ausweb.scu.edu.au/aw96/tech/aoun/paper.htm [April 5, 2005].

[Damiani, 2002] Damiani E., Vimercati S. C., Paraboschi S., and Samarati P., "A fine-grained access control system for XML documents.", Fifth ACM Workshop on Information and System Security, Vol.5, No.2, 2002

[Delis, 2005] Alex D., Lisa H., Nasir M., and Torsten S.., "Database and information retrieval. " Department of Computer and Communication Science, Polytechnic University, http://cis.poly.edu/research/dsb.shtml [April 5, 2005].

[Hong, 2004] Hong C.,"Role-based fine-grained XML access control", Masters thesis, Computer Science Department, Oklahoma State University, 2004

[Koch, 2002] Koch M., Mancini L. V., and Parisi-Presicce F., "A graph-based formalism for RBAC.", Fifth ACM Workshop on Information and System Security, Vol.5, No.3, 2002

[Park, 2001] Park J. S., Ahn G. J., and Shandu R., "Role-based access control on the web.", Fourth ACM Workshop on Information and System Security, Vol.4, No.1, 2001

[Ravi, 2000] Ravi S., David F., and Richard K., "The NIST model for role-based access control: towards a unified standard", ACM workshop on Role-based access control, Vol.5, No.6, 2000

[Wainer, 2005] Wainer J., and Kumar A., "A Fine-Grained, Controllable, User-toUser Delegation Method in RBAC", ACM workshop on Role-based access control, pages 59-66, ACM press, 2005.

[Workshop, 2005] Workshop on networked information retrieval, http://ciir.cs.umass.edu/nir96 [April 5, 2005].

[Yao, 2005] Yao J.T., Yao Y.Y., "Web-based information retrieval support systems: building research tools for scientists in the new information age, http://www2.cs.uregina.ca/~jtyao/Papers/113.pdf [April 5, 2005].

[Zhang , 2003] Zhang L., Ahn G. J.,"A rule-based framework for role-based delegation and revocation.", Sixth ACM Workshop on Information and System Security, Vol.6, No.3, 2003

VITA

Hongi Chandra Tjan

Candidate for the Degree of

Master of Science

Thesis: A GRAPH-BASED MECHANISM FOR BRIDGING ACCESS CONTROL LANGUAGE

Major Field: Computer Science

Biographical:

Personal Data: Born in Medan, Indonesia, On January 24, 1980, the son of Hong Tjai Khiang and Tjan Siu Tjoe.

Education: Graduated from Kristen Yusuf High School, Jakarta Utara, Indonesia in May 1998; received Bachelor of Science degree in Computer Science from Oklahoma State University, Stillwater, Oklahoma in May 2002. Completed the requirements for the Master of Science degree with major in Computer Science at Oklahoma State University in (May, 2006)

Experience: Employed by Oklahoma State University, Residential Life as a computer technician for on campus residence, Fall 1999; Employed by Designer Wicker and Rattan as the network specialist during summer 2001.

Professional Memberships: The Starr Foundation Scholarship, Golden Key, National Society Collegiate Scholars.

Name: Hongi Chandra Tjan Date of Degree: May, 2006

Institution: Oklahoma State University Location: Tulsa, Oklahoma

Title of Study: A COMBINED FINE-GRAINED AND ROLE-BASED ACCESS CONTROL MECHANISM

Pages in Study: 66 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: Different access control methods have been proposed to ensure data security in a computer system. Approaches to access control include role-based access control and fine-grained access control. However these systems suffer from complexity or inadequate security. Although the role-based access control is efficient in terms of the overheads related to security it is not as secure as fine-grained access control. On the other hand fine-grained access control is secure, but very inefficient for storing access information. To solve this problem, we propose a combined fine grained-role based access control system.
A graph representation is used to capture the combined model. Furthermore, in this thesis we propose a combined system which caters for the following:

- System with one user and one role
- System with one user and multiple roles
- System with multiple users and one role
- System with multiple users and multiple roles.

Findings and Conclusions: Formal Graph Merging operations for the above four scenarios have been defined. The merging operation merges a graph representing the fine-grained system with a graph representing the role based system to generate a new graph model of the combined role-based fine-grained system. The combined system introduces the new rules for access control based on the above four categories. Simulation results show that the combined system has the efficiency of the role-based access control and at the same time, the security of the fine-grained control system. . A formal grammar in introduced to capture the access control for the combined system. Future works will involve implementing the proposed system in a real world environment.

ADVISER'S APPROVAL: Dr. Johnson P. Thomas