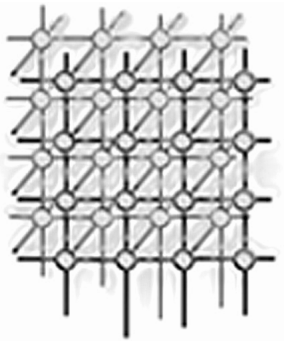
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2000; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*



Specifying and Enforcing the Principle of Least Privilege in Role-Based Access Control

Xiaopu Ma[1,2],Ruixuan Li[1,*,†], Zhengding Lu[1],
Jianfeng Lu[1]and Meng Dong[1]

[1] *College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, P.R.China*
[2] *College of Computer and Information Technology, Nanyang Normal University, Nanyang 473061, Henan, P.R.China*

SUMMARY

The principle of least privilege in role-based access control (RBAC) is an important area of research. There are two crucial issues related to it: the specification and the enforcement. We believe that existing least privilege specification schemes are not comprehensive enough and few of the enforcement methods are likely to scale well. In this paper, we formally define the basic principle of least privilege and present different variations, called the *δ-approx* principle of least privilege and the *minimizing-approx* principle of least privilege. Since there may be more than one result to enforce the same principle of least privilege, we introduce the notation about weights of permission and role to optimize the results. Then we prove that all least privilege problems are NP-complete. As an important contribution of the paper, we show the principle of least privilege problem can be reduced to minimal cost set covering (MCSC) problem. We can borrow the existing solutions of MCSC to solve the principle of least privilege problems. Finally, different algorithms are designed to solve the proposed least privilege problems. Experiments on performance study prove the superiority of our algorithms.

KEY WORDS: role-based access control; principle of least privilege; weight; enforcement

1. INTRODUCTION

Role-based access control (RBAC) [1] is the most popular access control model at present, and is widely used as an alternative to the traditional discretionary access control (DAC) and the

*Correspondence to: College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, P.R.China
†E-mail: rxli@hust.edu.cn

Received
Revised

mandatory access control (MAC) in enterprise security management products. In RBAC, a set of permissions are assigned to users through roles. This change on how to assign the permissions often reduces the complexity of access control because the number of users is generally much larger than that of roles in an organization. The most distinctive and important feature of the RBAC is the desire to specify and enforce enterprise-specific security policies in a way that maps naturally to an organization's structure [2]. Its emphasis on controlling who has access to operations on what objects is fundamentally different from information flow security in multi-level secure systems, and therefore can support three well-known security principles: least privilege, separation of duties and data abstraction [3].

The principle of least privilege is one of the most important principles in the design of protection mechanisms for secure computer systems [4]. Whenever possible, a user should be given no more access to resources than it is required to complete the task at hand. In other words, the computer system should be able to determine the minimum set of privileges required for the user to perform the task and guarantee that the user is only granted those privileges and no more. Chen and Jason [5] proposed a set covering optimization method to enforce the principle of least privilege under a family of simple RBAC models. In [6], Schneider developed the principle of least privilege in connection with devising security enforcement mechanisms for systems structured in terms of base and a set of extensions which augment the functionality of that base. Li et al. [7, 8] provided a method to enforce the principle of least privilege in multi-domain environments.

While solving the basic principle of least privilege problem, the goal is to identify the minimal set of roles whose permissions exactly equal to the requested permissions set. However, in real systems there may not be such a situation for the requested permissions set. Let us assume that the requested permissions set $RQ = \{p_1, p_2, p_3, p_4\}$, $auth\_perms(r_1) = \{p_1, p_2\}$ (permissions p_1, p_2 belong to role r_1), $auth\_perms(r_2) = \{p_3, p_4, p_5\}$ (permissions p_3, p_4, p_5 belong to role r_2), $auth\_perms(r_3) = \{p_1, p_2, p_3\}$ (r_3 has the permissions p_1, p_2 and p_3). In this situation, there is no minimal set of roles whose permissions exactly equal to the requested permission set RQ. However, if we allow a slight redundant permissions rather than the exactly requested permissions, it may still be acceptable. For example, in this situation we can give the redundant permission p_5 to the user. It is this variation of the principle of least privilege that we recognize as the δ-*approx* principle of least privilege. Moreover, there may be a cardinality constraint or other constraints on the roles which belong to one user [9, 10]. In this situation, there also may not generate the role set that the union permissions belong to these roles exactly equal to the requested permissions set. This discrepancy is recognized as the *minimizing-approx* principle of least privilege. For example, each user have not more than $limit(r)$ roles, however there may not be $limit(r)$ roles' permissions that exactly equal to the requested permissions set.

Furthermore, since more than one role set may satisfy the different principle of least privilege problems. For example, role set $\{r_1, r_2\}$ and $\{r_3, r_4\}$ both enforce the principle of least privilege for the same requested permission set RQ, how can we assess the difference between them? which is the better result for the administrator to choose? In this context, we introduce the notation about the weight of permission and role to optimize these problems. In practice, the weight of permission is a value attached to a permission representing its importance. Depending on the domain, there could be any variable ranging from the types of operations to the types of objects. For example, the permission "read" of the patient's personal information may be

more important than the permission "write" to the patient's personal information. This is because the "read" permission usually leads to more information leakage. In another case, the permission "write" to the students' achievement may be more significant that the permission "read" of the students' achievement. Hence we will choose the result with lesser weight if there are more than one role set to enforce the same principle of least privilege.

In this paper, we present a unified definition to describe the basic principle of least privilege problem and show that the decision version of the problem is NP-complete. We also consider several interesting variations of the basic principle of least privilege problem, including the *δ-approx* principle of least privilege and the *minimizing-approx* principles of least privilege problem. These are of practical importance while both the *δ-approx* principle of least privilege and the *minimizing-approx* principle of least privilege are likely to result in more permissions than the basic principle of least privilege. Furthermore, we introduce the notion about the weight of permission and role to optimize the results of the least privilege problems.

We have discovered that the basic principle of least privilege problem is identical to the minimal cost set covering (MCSC) problem [11]. This fact allows us to directly adopt many heuristic solutions and tools developed for the MCSC problem to solve the basic principle of least privilege problem. Although the MCSC solutions are applicable to any domain of interest, in this paper, we present the MCSC problem in a least privilege context in order to enforce the basic principle of least privilege problem. And we also propose different algorithms to solve the variants of the basic principle of least privilege problem. The experiments are carried out to evaluate the performance of our algorithms and the results prove its superiority.

The rest of this paper is organized as follows. Section 2 reviews the RBAC model and some preliminary definitions employed in this paper. Section 3 defines the basic principle of least privilege as well as its variants. Section 4 analyzes the computational complexity of the proposed variant principles of least privilege problems. In Section 5, we map the basic principle of least privilege problem to the minimal cost set covering problem and give different algorithms to solve its variants. The performance of all the algorithms is presented in Section 6. Finally, Section 7 provides some insights into our ongoing and future work.

2. PRELIMINARIES

We develop the material in this paper in the context of RBAC 96, the most widely known role-based access control model [1] (it consists of RBAC0, RBAC1, RBAC2 and RBAC3, the last two of which incorporate separation of duty constraints. For the sake of simplicity, we do not consider sessions in this paper).

Definition 1.(RBAC Model) The RBAC model has the following components:

- U, R, P, users, roles and permissions respectively;
- $PA \subseteq P \times R$, a many-to-many mapping of permission to role assignments;
- $UA \subseteq U \times R$, a many-to-many user to role assignment relationships;
- $auth\_perm(R) = \{p \in P | (p, R) \in PA\}$, a set of permissions that a role has.

We can use a $m \times n$ binary matrix M to describe the relationships between roles and permissions where m is the number of roles and n is the number of permissions. The element

$M\{\text{i,j}\}$=1 denotes that the ith role has the jth permission or the jth permission belongs to the ith role. In this situation, each row indicates an n-dimensional role vector and also indicates what permissions belong to the role. We can use r_i (i=1,...m) to indicate the ith role, p_j (j=1,...n) to indicate the jth permission. We can give some operations to n-dimensional boolean role vector as follows.

Definition 2.(N-dimensional Boolean Role Vector Substraction Operator) An n-dimensional boolean role vector substraction operation between n-dimensional boolean role vector $A \in \{0,1\}^n$ and $B \in \{0,1\}^n$ is $A - B = C$ where C is in space Z^n (where Z is integer) and

$$c_j = a_j - b_j \ (j = 1, ...n)$$

Definition 3.(N-dimensional Boolean Role Vector Logical OR Operator) An n-dimensional boolean role vector logical OR operation between n-dimensional boolean role vector $A \in \{0,1\}^n$ and $B \in \{0,1\}^n$ is $A||B = C$ where C is in space $\{0,1\}^n$ and

$$c_j = a_j|b_j \ (j = 1, ...n)$$

So

$$r_1 \cup r_2 \cup ... \cup r_k = (M\{1,1\}|...|M\{k,1\}, M\{1,2\}|...|M\{k,2\}, ...M\{1,n\}|...|M\{k,n\})$$

where $k \in$ N and $1 \leq k \leq m$.

Definition 4. (L_1-Metric) The distance metric between n-dimensional boolean role vector $A \in \{0,1\}^n$ and $B \in \{0,1\}^n$ is defined as

$$\parallel A - B \parallel_1 = \sum_{j=1}^{n} |a_j - b_j|$$

The L_1-Metric can be used to count the difference between two n-dimensional boolean role vectors-i.e., to figure out how good an approximation one is of the other. When the L_1-Metric is 0, the two n-dimensional boolean role vectors are identical.

We now introduce the notion of δ-consistency between the role set $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ (each role in R) and the requested permissions set $RQ \subseteq P$ (we use an n-dimensional boolean vector Q to describe the requested permissions set RQ , where $q_j = 1$ if $p_j \in RQ$ or $q_j = 0$ if $p_j \notin RQ$ (j=1,...n)). And we also give a definition about redundancy permissions.

Definition 5.(δ-Consistency) Given a set of roles $r_1, r_2...r_k$ and a set of requested permissions RQ, $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ is δ-consistency with RQ if and only if

$$\parallel r_1 \bigcup r_2 \bigcup ... \bigcup r_k - Q \parallel_1 \leq \delta$$

Definition 6.(Redundancy Permissions) Given a set of roles $r_1, r_2...r_k$ and a set of requested permissions RQ, the redundancy permissions about $r_1, r_2...r_k$ to RQ is defined as

$$\mathrm{RP}(r_1, r_2...r_k; RQ) = \{p \in \bigcup_{i=1}^{k} auth\_perm(r_i) | p \notin RQ\}$$

In practice, the weight of permission is a value attached to a permission representing its importance. We denote it as ω_p. Depending on the domain, there could be any variable ranging

from the types of operations to the types of objects. In other words, the weight of permission is a function of selected weighting attributes therefore denoted as $\omega_p = f(a_1, a_2, ...a_n)$ (where a_i (i=1,...n) is the selected weighting attributes). For the sake of simplicity, we can define the weight of permission as follows.

Definition 7.(Weight of Permission) The weight of permission p_i is defined as

$$w_{p_i} = \alpha \times f_1(ua) + \beta \times f_2(opa) + \gamma \times f_3(oba) + \delta \times w_0$$

where ω_0 is the initial weight of permission p_i preset by the system based on the knowledge of comprehensive effect of all factors on permission p_i, $f_1(ua)$ describes the effect of the user's attributes on the weight of permission, $f_2(opa)$ describes the effect of the operation's attributes on the weight of permission and $f_3(oba)$ describes the effect of the object's attributes on the weight of permission; α, β, γ and δ are parameters used to adjust the relative importance about the weight of permission p_i corresponding to each attribute. If we have no prior knowledge of the initial weight, we can set δ to 0. The idea behind the definition based on the theory described in [12].

We assign a real number $\omega_{p_i} \in [0, 1]$ to p_i for each permission $p_i \in P$ (i=1,...n), which we call the weight of permission that can be calculated by the Definition 7. Since the role is a set of permissions, we must define weights for all roles. This can be done as follows.

For any $r_i \in R$ (i=1,...m), suppose $auth\_perm(r_i) = \{p_1, p_2, ...p_k\}$, where $p_j \in P$ ($j = 1, ...k$), we define the weight of r_i as follows.

Definition 8.(Weight of role) The weight of role r_i is defined as

$$\omega_{r_i} = \mathrm{F}(\omega_{p_1}, \omega_{p_2}, ..., \omega_{p_k}) \quad \text{where} \quad \mathrm{F} : \mathrm{R}^k \to [0, 1]$$

Hence we can get the definition about the weight of redundancy permissions and the weight of role set.

Definition 9. (Weight of Redundancy Permissions) Given a set of roles $r_1, r_2...r_k$, a set of requested permissions RQ and a set of weight ω_{p_j} for each permission $p_j \in P$ (j=1,...n), the weight of redundancy permissions about $r_1, r_2...r_k$ to RQ is defined as

$$\mathrm{WRP}(r_1, r_2, ...r_k; RQ) = \{\mathrm{F}'(\omega_{p_j}) | p_j \in \mathrm{RP}(r_1, r_2, ...r_k; RQ)\} \text{ where } \quad \mathrm{F}' : \mathrm{R}^{|\mathrm{RP}|} \to [0, 1]$$

Definition 10. (Weight of Role Set) Given a set of roles $r_1, r_2...r_k$ and a set of weight ω_{r_i} (i=1,...k) for each role, the weight of role set about $r_1, r_2...r_k$ is defined as

$$\mathrm{WRS}\{r_1, r_2, ...r_k\} = \frac{\sum_{i=1}^{k} \omega_{r_i}}{k}$$

3. The Basic Principle of Least Privilege and Its Variants

In this section, we give formal definition for the basic principle of least privilege and its variants respectively.

3.1. The Basic Principle of Least Privilege

The basic principle of least privilege provides a protection for the security computer systems which ensures that a user should be given no more access to resources than is required to complete the task at hand. So we formally present the definition of the basic principle of least privilege as follows.

Definition 11. (The Basic Principle of Least Privilege(BLP)) Given a set of requested permissions RQ, find the minimal number set of roles k such that $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ 0-consistency with RQ.

Given the requested permissions set RQ, we have to find the minimal number set of roles that can exactly cover all of the permissions in the requested permissions set RQ. However, the results of minimal number set of roles may not be only one set. That is, there may have several sets $r_1, r_2...r_k$ such that $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ 0-consistency with RQ and all k is minimal. While more than one set of roles may all enforce the basic principle of least privilege for the same requested permissions set, they may have different weights.

Definition 12. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of roles that both enforce the basic principle of least privilege for the same requested permissions set. We say that the weight of $\mathcal{C}_1$ is at least as big as that of $\mathcal{C}_2$ (denoted by $\mathcal{C}_1 \succeq \mathcal{C}_2$) if, and only if, the following holds:

$$\mathrm{WRS}\{\mathcal{C}_1\} \succeq \mathrm{WRS}\{\mathcal{C}_2\}$$

Then $\succeq$ relation among all sets of roles that enforce the same basic principle of least privilege is a partial order. When $\mathcal{C}_1 \succeq \mathcal{C}_2$, but not $\mathcal{C}_2 \succeq \mathcal{C}_1$, we say that the weight of $\mathcal{C}_1$ is more than $\mathcal{C}_2$(denoted by $\mathcal{C}_1 \succ \mathcal{C}_2$). By definition, $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent if, and only if, $\mathcal{C}_1 \succeq \mathcal{C}_2$ and $\mathcal{C}_2 \succeq \mathcal{C}_1$.

When both $\mathcal{C}$ and $\mathcal{C}'$ enforce the basic principle of least privilege for the same requested permissions set requirements, there are three cases:

1. $\mathcal{C} \succ \mathcal{C}'$, in which case, $\mathcal{C}'$ is preferable to $\mathcal{C}$ for enforcing the basic principle of least privilege for the same requested permissions set as it has lesser weight or lesser harmful;
2. $\mathcal{C}' \succ \mathcal{C}$, in which case, $\mathcal{C}$ is preferable to $\mathcal{C}'$;
3. $\mathcal{C}$ and $\mathcal{C}'$ are equivalent, in which case, either $\mathcal{C}$ or $\mathcal{C}'$ can be used.

Then we can define the optimal basic principle of least privilege based on the weight of role set as follows.

Definition 13. (The Optimal Basic Principle of Least Privilege(OBLP)) Given a set of requested permissions RQ, we say that a set $\mathcal{C}$ of roles is optimization for enforcing the basic principle of least privilege for the requested permissions set RQ if, and only if, $\mathcal{C}$ enforces the basic principle of least privilege for the requested permissions set RQ and there does not exist a different set $\mathcal{C}'$ of roles such that $\mathcal{C}'$ also enforces the basic principle of least privilege for the same requested permissions set RQ and $\mathcal{C} \succ \mathcal{C}'$ (the weight of $\mathcal{C}'$ is less than the weight of $\mathcal{C}$).

3.2. Variants of the Basic Principle of Least Privilege

While exact match is a good thing to have, approximate match might be a prudent choice for the real systems. For example, if there are many casual users from the outside want to complete tasks. We exactly know each user's permissions in this situation, but the minimal number set of roles may not exist, that is, there may not be a set of roles $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ such that $r_1 \bigcup r_2 \bigcup ... \bigcup r_k$ 0-consistency with the casual user's requested permissions set. Then there may be three approaches to solve this problem:

1. The administrator can not give any role to the casual user because there is no roles set which exactly matches the casual user's requested permissions set, hence there is no operability for the casual user in this way. In other words, the casual user can not complete the task at hand;
2. The administrator can give a set of roles to the casual user through altering role hierarchies to satisfy the casual user's requested permissions set. In this case, there are three disadvantages. Firstly, role hierarchies reflect, to some extent, the applications' organizational structures, which is the important characteristic of RBAC. With role hierarchies not conforming to the organizational structures, RBAC would be less attractive; Secondly, the roles may be split many times and many new roles would appear. If there are a large number of casual users, the role hierarchies will be too complicated to be understood; Last but not least, it would incur large administration overhead [8] and
3. The administrator directly assigns each requested permission to the casual users. However, this approach is cumbersome and makes it difficult for security administrator to manage. On the other hand, this approach can not utilize the benefits of RBAC.

In this situation, if we want to conveniently provide a role set for the user to complete the work, the notion of δ-consistency is useful, since it helps to bound the degree of approximation. Therefore, we define the approximate principle of least privilege using δ-consistency as follows.

Definition 14. (The δ-approx Principle of Least Privilege (δLP)) Given a set of requested permissions RQ, and a threshold $\delta \geq 0$, find a set of roles $r_1, r_2...r_k$ that δ-consistency with RQ and minimizing the number of roles, k.

It should be made clear that the basic principle of least privilege defined earlier is simply a special case of the δ-approx principle of least privilege (with δ set to 0). In this principle of least privilege, given the same threshold δ, there may be many sets of roles which are δ-consistency with the requested permissions RQ and all k is minimal. Then a measurement is needed to make sure how good a set of roles are, so that they can be selected among.

Definition 15. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of roles that both enforce the same δ-approx principle of least privilege with the requested permissions set RQ. We say that the weight of redundancy permissions of $\mathcal{C}_1$ to the requested permissions set RQ is at least as big as the weight of redundancy permissions of $\mathcal{C}_2$ to the requested permissions set RQ (denoted by $\mathcal{C}_1 \succeq \mathcal{C}_2$) if, and only if, the following holds:

$$\mathrm{WRP}(\mathcal{C}_1; RQ) \succeq \mathrm{WRP}(\mathcal{C}_2; RQ)$$

Then $\succeq$ relation among all sets of roles that enforce the same δ-approx principle of least privilege with the requested permissions set RQ is a partial order. When $\mathcal{C}_1 \succeq \mathcal{C}_2$, but not $\mathcal{C}_2 \succeq \mathcal{C}_1$, we say that the weight of redundancy permissions of $\mathcal{C}_1$ to the requested permissions set RQ is more than the weight of redundancy permissions of $\mathcal{C}_2$ to the requested permissions set RQ (denoted by $\mathcal{C}_1 \succ \mathcal{C}_2$). By definition, $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent if, and only if, $\mathcal{C}_1 \succeq \mathcal{C}_2$ and $\mathcal{C}_2 \succeq \mathcal{C}_1$.

When both $\mathcal{C}$ and $\mathcal{C}'$ enforce the same δ-approx principle of least privilege with the requested permissions set RQ, there are three cases:

1. $\mathcal{C} \succ \mathcal{C}'$, in which case, $\mathcal{C}'$ is preferable to $\mathcal{C}$ for enforcing the same δ-approx principle of least privilege with the requested permissions set RQ as it has lesser weight of redundancy permissions or lesser harmful to the system when the casual user gets more permissions than required;
2. $\mathcal{C}' \succ \mathcal{C}$, in which case, $\mathcal{C}$ is preferable to $\mathcal{C}'$;
3. $\mathcal{C}$ and $\mathcal{C}'$ are equivalent, in which case, either $\mathcal{C}$ or $\mathcal{C}'$ can be used.

Then we can define the optimal δ-approx principle of least privilege based on the weight of redundancy permissions to the requested permissions set RQ as follows.

Definition 16. (The Optimal δ-approx Principle of Least Privilege(OδLP)) Given a set of requested permissions RQ, we say that a set $\mathcal{C}$ of roles is optimization for enforcing the δ-approx principle of least privilege for the requested permissions set RQ if, and only if, $\mathcal{C}$ enforces the δ-approx principle of least privilege for the requested permissions set RQ and there does not exist a different set $\mathcal{C}'$ of roles such that $\mathcal{C}'$ also enforces the δ-approx principle of least privilege for the requested permissions set RQ and $\mathcal{C} \succ \mathcal{C}'$. (the weight of redundancy permissions of $\mathcal{C}'$ is less than the weight of redundancy permissions of $\mathcal{C}$).

Instead of bounding the approximation, and minimizing the number of roles, it might be interesting to do the reverse-bound the number of roles, and minimizing the approximation. This will occur once that there are cardinality constraints in RBAC systems. For example, each user can hold not more than three roles. In this situation, there are not more than three roles 0-consistency with a set of requested permissions. Then we need define the minimizing-approx principle of least privilege as follows.

Definition 17. (The Minimizing-approx Principle of Least Privilege(MLP)) Given a set of requested permissions RQ, and the number of roles k, find a set of roles not more than k such that $r_1, r_2...r_j$ $(j \leq k)$ δ-consistency with the requested permissions set RQ and minimizing the δ.

Similarly, the results of minimizing-approx principle of least privilege for the requested permissions set RQ may be more than one set. We can also use the same measurement described in the δ-approx principle of least privilege in order to select the optimal set among them. And the optimal minimizing-approx principle of least privilege is described as follows.

Definition 18. (The Optimal Minimizing-approx Principle of Least Privilege(OMLP)) Given a set of requested permissions RQ, we say that a set $\mathcal{C}$ of roles is optimization for enforcing the minimizing-approx principle of least privilege for the requested permissions set RQ if, and only if, $\mathcal{C}$ enforces the minimizing-approx principle of least privilege for the requested permissions set RQ and there does not exist a different set $\mathcal{C}'$ of roles such that

Table I. A sample role-permission assignments for describing the basic principle of least privilege

	p_1	p_2	p_3	p_4	p_5	p_6
r_1	1	1	0	0	0	0
r_2	0	0	1	1	0	0
r_3	1	0	1	0	0	0
r_4	0	1	0	1	0	0
r_5	1	1	0	0	1	0
r_6	0	0	0	0	1	1

$\mathcal{C}'$ also enforces the minimizing-approx principle of least privilege for the requested permissions set RQ and $\mathcal{C} \succ \mathcal{C}'$ (the weight of redundancy permissions of $\mathcal{C}'$ is less than the weight of redundancy permissions of $\mathcal{C}$).

We can clarify these problems further by means of three examples. Table I shows a sample role-permission assignments, for 6 roles and 6 permissions. Let us assume the requested permissions set $RQ = \{p_1, p_2, p_3, p_4\}$, then $\{r_1, r_2\}$ and $\{r_3, r_4\}$ are both minimal role sets that 0-consistency with $\{p_1, p_2, p_3, p_4\}$.

There are three situations for the administrator to choose in order to satisfy the basic principle of least privilege for the requested permissions set:

1. **WRS$\{r_1, r_2\}$ > WRS$\{r_3, r_4\}$:** We would like to choose the role set $\{r_3, r_4\}$ for enforcing the basic principle of least privilege for the requested permissions set $\{p_1, p_2, p_3, p_4\}$ in order to decrease the risk of harming the security systems because the role set $\{r_3, r_4\}$ has lesser weight or being less harmful (Here we can incorporate risk to the weight of each permission and role in order to reflect the harmfulness to the security systems);
2. **WRS$\{r_1, r_2\}$ < WRS$\{r_3, r_4\}$:** Obviously, we would like to choose the role set $\{r_1, r_2\}$ for enforcing the basic principle of least privilege in order to decrease the risk of harming the security systems;
3. **WRS$\{r_1, r_2\}$ = WRS$\{r_3, r_4\}$:** In this situation, both the role set $\{r_1, r_2\}$ and $\{r_3, r_4\}$ can be chosen for enforcing the basic principle of least privilege.

Table II shows a sample role-permission assignments, for 5 roles and 6 permissions. Let us assume the requested permissions set $RQ = \{p_1, p_2, p_3, p_4\}$, the weight of each permission is $\omega_{p_j} \in [0, 1]$ $(j = 1, ...6)$. Obviously there is no role set which exactly matches the requested permissions set RQ, hence we choose $\delta = 1$ and k=2 to describe the 1-approx principle of least privilege (In the δ-approx principle of least privilege problem, we want to find the minimal number of roles to satisfy the 1-consistency with the requested permissions set RQ, here the minimal number of roles is 2 when $\delta = 1$).

Then there are five cases that satisfy the 1-approx principle of least privilege for the requested permissions set RQ and the minimum number of k is 2:

1. $\{r_1, r_2\}$ is 1-consistency with the requested permissions set RQ and the minimum number of k is 2, in which case, RP$(r_1, r_2; RQ) = \{p_5\}$,WRP$(r_1, r_2; RQ) =$ F$'(\omega_{p_5})$;

Table II. A sample role-permission assignments for describing the 1-approx principle of least privilege

	p_1	p_2	p_3	p_4	p_5	p_6
r_1	1	1	0	0	0	0
r_2	0	0	1	1	1	0
r_3	1	1	1	0	0	0
r_4	0	1	0	1	1	0
r_5	0	0	1	1	0	1

2. $\{r_1, r_5\}$ is 1-consistency with the requested permissions set RQ and the minimum number of k is 2, in which case, $\mathrm{RP}(r_1, r_5; RQ) = \{p_6\}$,$\mathrm{WRP}(r_1, r_5; RQ) = \mathrm{F}'(\omega_{p_6})$;
3. $\{r_2, r_3\}$ is 1-consistency with the requested permissions set RQ and the minimum number of k is 2, in which case, $\mathrm{RP}(r_2, r_3; RQ) = \{p_5\}$,$\mathrm{WRP}(r_2, r_3; RQ) = \mathrm{F}'(\omega_{p_5})$;
4. $\{r_3, r_4\}$ is 1-consistency with the requested permissions set RQ and the minimum number of k is 2, in which case, $\mathrm{RP}(r_3, r_4; RQ) = \{p_5\}$,$\mathrm{WRP}(r_3, r_4; RQ) = \mathrm{F}'(\omega_{p_5})$;
5. $\{r_3, r_5\}$ is 1-consistency with the requested permissions set RQ and the minimum number of k is 2, in which case, $\mathrm{RP}(r_3, r_5; RQ) = \{p_6\}$,$\mathrm{WRP}(r_3, r_5; RQ) = \mathrm{F}'(\omega_{p_6})$;

Here case 1, case 3 and case 4 are equivalent because the weight of redundancy permission are all $\mathrm{F}'(\omega_{p_5})$, case 2 and case 5 are equivalent because the weight of redundancy permission are both $\mathrm{F}'(\omega_{p_6})$. There are three situations for the administrator to choose in order to satisfy the 1-approx principle of least privilege for the requested permissions set:

1. $\mathbf{F}'(\boldsymbol{w_{p_5}}) > \mathbf{F}'(\boldsymbol{w_{p_6}})$: In this situation, we believe the redundancy permission p_5 is more important than p_6. Hence, case 2 and case 5 are preferable to case 1, case 3 and case 4 for enforcing the same 1-approx principle of least privilege. In fact, it is also true in the real RBAC systems if there is no exactly role set which satisfies the requested permissions set, it may be an advisable method for allocating the unimportant redundant permissions to the casual user to complete the task without bringing serious harm to the security systems;
2. $\mathbf{F}'(\boldsymbol{w_{p_6}}) > \mathbf{F}'(\boldsymbol{w_{p_5}})$: In this situation, the redundancy permission p_6 is more important than p_5. Hence case 1, case 3 and case 4 are preferable to case 2 and case 5 for enforcing the same 1-approx principle of least privilege for the requested permissions set RQ because they have the lesser weight of redundancy permissions;
3. $\mathbf{F}'(\boldsymbol{w_{p_6}}) = \mathbf{F}'(\boldsymbol{w_{p_5}})$: In this situation, the redundancy permission p_6 is as important as p_5 . Hence case 1, case 3 and case 4 are equal to case 2 and case 5 for enforcing the same 1-approx principle of least privilege for the requested permissions set.

Table III shows a role-permission assignments for the minimizing-approx principle of least privilege problem where there are 6 roles and 6 permissions. Let us assume the requested permissions set $RQ = \{p_1, p_2, p_3, p_4\}$, the weight of each permission is $\omega_{p_j} \in [0, 1]$ $(j = 1, ...6)$. Obviously there is not an exact match for the requested permissions set, so we choose k=2 to describe the minimizing-approx principle of least privilege (In the minimizing-approx principle

Table III. A sample role-permission assignments for the minimizing-approx principle of least privilege

	p_1	p_2	p_3	p_4	p_5	p_6
r_1	1	1	1	0	0	0
r_2	0	0	0	1	1	1
r_3	1	1	0	0	0	0
r_4	0	0	1	1	1	0
r_5	0	0	0	1	1	0
r_6	0	0	0	0	1	1

of least privilege problem, we want to find no more than k roles to satisfy the δ-consistency with the requested permissions set RQ and minimizing the δ at the same time, here we choose the number of roles is 2).

Then there are four cases that satisfy the minimizing-approx principle of least privilege for the requested permissions set:

1. $\{r_1, r_2\}$ is 2-consistency with the requested permissions set RQ and the number of roles is no more than 2, in which case, $\mathrm{RP}(r_1, r_2; RQ) = \{p_5, p_6\}$,$\mathrm{WRP}(r_1, r_2; RQ) = \mathrm{F}'(\omega_{p_5}, \omega_{p_6})$;
2. $\{r_1, r_4\}$ is 1-consistency with the requested permissions set RQ and the number of roles is no more than 2, in which case, $\mathrm{RP}(r_1, r_4; RQ) = \{p_5\}$,$\mathrm{WRP}(r_1, r_4; RQ) = \mathrm{F}'(\omega_{p_5})$;
3. $\{r_1, r_5\}$ is 1-consistency with the requested permissions set RQ and the number of roles is no more than 2, in which case, $\mathrm{RP}(r_1, r_5; RQ) = \{p_5\}$,$\mathrm{WRP}(r_1, r_5; RQ) = \mathrm{F}'(\omega_{p_5})$;
4. $\{r_3, r_4\}$ is 1-consistency with the requested permissions set RQ and the number of roles is no more than 2, in which case, $\mathrm{RP}(r_3, r_4; RQ) = \{p_5\}$,$\mathrm{WRP}(r_3, r_4; RQ) = \mathrm{F}'(\omega_{p_5})$;

If we choose the sum as the function to compute the weight of the redundancy permissions, according to Definition 18, case 2, case 3 and case 4 are all preferable to case 1 because the redundancy permission's weight is lesser. Case 2, case 3 and case 4 are equivalent because they have the same weight of redundancy permission.

4. Computational Complexity

The basic principle of least privilege and its variants are all optimization problems. The theory of NP-completeness applies to decision problems. Therefore, in order to consider the complexity of the problems, we now give the decision version of these problems.

Definition 19.(Decision BLP) Given a set of requested permissions RQ, and the number of roles k, are there a set of roles, $ROLES$, 0-consistency with the requested permissions set RQ such that $|ROLES| \leq k$ is called the basic principle of least privilege decision problem.

Definition 20.(Decision δLP) Given a set of requested permissions RQ, the number of roles k, and a threshold $\delta \geq 0$, are there a set of roles, $ROLES$, δ-consistency with the requested

permissions set RQ such that $|ROLES| \leq k$ is called the δ-approx principle of least privilege decision problem.

Definition 21.(Decision MLP) Given a set of requested permissions RQ, the number of roles k, and an approximate threshold $\theta \geq 0$, are there a set of roles, $ROLES$, such that

$$\| r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \|_1 \leq \theta \ (1 \leq j \leq k)$$

is called the minimizing-approx principle of least privilege decision problem.

We now reduce the known minimum cover problem to these problems to show that the above decision problems are all NP-complete problems [13].

Definition 22.(Minimum Cover Problem) Given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{S}$, and a positive integer $k \leq |\mathcal{C}|$, is there a collection B of subsets $\mathcal{S}$ such that $|B| \leq k$ whose union is exactly $\mathcal{S}$?

Theorem 1. The decision BLP is NP-complete.

Proof. We first show that decision BLP problem is in NP. If one correctly guesses the set of roles, verifying that the guess is correct that can be done in polynomial time: compute the union of the roles' permissions and check whether it is equal to the set of permissions in RQ, compute the number of roles and check whether it is not more than k. Therefore, the decision BLP problem is in NP.

The transformation is quite simple. Given an instance of the minimum cover problem, here is how we transform it to an instance of the decision BLP problem: $\mathcal{S}$ denotes the requested permissions set RQ, each $c \in \mathcal{C}$ denotes the role. Now, the answer to the decision BLP problem directly provides the answer to the minimum cover problem (Obviously, the transformation is polynomial). □

Theorem 2. The decision δLP is NP-complete.

Proof. We first show that the δLP decision problem is in NP. If one correctly guesses the set of roles $r_1, r_2...r_j$, it only takes polynomial time to compute

$$\| r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \|_1$$

and ensure that it is less than or equal to δ, and that the number of roles j is not more than k. Therefore, the δLP decision problem is in NP.

The transformation from the δLP decision problem to the minimum cover problem is quite simple. Given an instance of the minimum cover problem, here is how we transform it to an instance of the decision δLP problem: $\mathcal{S}$ denotes the set of requested permissions RQ, each $c \in \mathcal{C}$ denotes the role, and set $\delta = 0$. Now the δLP decision problem is enforceable if and only if not more than k sets in $\mathcal{C}$ cover $\mathcal{S}$ (Obviously, the transformation is polynomial). □

Theorem 3. The decision MLP is NP-complete.

Proof. We first show that the decision MLP problem is in NP. If one correctly guesses the set of roles $r_1, r_2...r_j$, it only takes polynomial time to compute

$$\| r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \|_1$$

and ensure that it is less than or equal to θ, and the number of roles j is not more than k. Therefore, the decision MLP problem is in NP.

The transformation from the decision MLP problem to the minimum cover problem is also quite simple. The transformation is as follows: $\mathcal{S}$ denotes the set of requested permissions RQ, each $c \in \mathcal{C}$ denotes the role, and set $\theta = 0$. Now the answer to the decision MLP problem directly provides the answer to the minimum cover problem (Obviously, the transformation is polynomial). □

Now we will define the decision version of the optimal basic principle of least privilege problem, the optimal δ-approx principle of least privilege problem, and the optimal minimizing-approx principle of least privilege problem as follows.

Definition 23.(Decision OBLP) Given a set of requested permissions RQ, the number of roles k, and the value of cost ω, are there a set of roles, $ROLES$, 0-consistency with the requested permissions set such that $|ROLES| \leq k$ and $\mathrm{WRS}\{ROLES\} \leq \omega$ is called the optimal basic principle of least privilege decision problem.

Definition 24.(Decision δLP) Given a set of requested permissions RQ, the number of roles k, the threshold $\delta \geq 0$, and the value of cost ω, are there a set of roles, $ROLES$, δ-consistency with the requested permissions set such that $|ROLES| \leq k$ and $\mathrm{WRP}\{ROLES; RQ\} \leq \omega$ is called the optimal δ-approx principle of least privilege decision problem.

Definition 25.(Decision OMLP) Given a set of requested permissions RQ, the number of roles k, the approximate threshold $\theta \geq 0$, and the value of cost ω, are there a set of roles, $ROLES$, such that $\mathrm{WRP}\{ROLES; RQ\} \leq \omega$ and

$$\| r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \|_1 \leq \theta \ (1 \leq j \leq k)$$

is called the optimal minimizing-approx principle of least privilege decision problem.

Corollary 1. The decision OBLP is NP-complete.

Proof. We first show that decision OBLP problem is in NP. If one correctly guesses the set of roles, verifying that the guess is correct that can be done in polynomial time: compute the union of the roles' permissions and check whether it is equal to the set of permissions in RQ, compute the number of roles and check whether it is not more than k and calculate the weight of these roles and check whether it is not more than ω. Therefore, the decision OBLP problem is in NP.

To prove the NP-hard, we consider a subclass of the problem-the basic principle of least privilege problem. According to the Theorem 1, we come to the conclusion that the basic principle of least privilege decision problem is NP-hard. Hence the decision OBLP is NP-complete. □

Corollary 2. The decision OδLP is NP-complete.

Proof. We first show that decision OδLP problem is in NP. If one correctly guesses the set of roles $r_1, r_2..r_j$, it only takes polynomial time to compute

$$\| r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \|_1$$

and ensure that it is less or equal to δ, and that the number of roles j is not more than k where the weight of these redundancy permissions is not more than ω. Therefore, the decision OδLP problem is in NP.

To prove the NP-hard, we consider a subclass of the problem-the basic δ-approx principle of least privilege problem. According to the Theorem 2, we come to the conclusion that the basic δ-approx principle of least privilege decision problem is NP-hard. Hence the decision OδLP is NP-complete. □

Corollary 3. The decision OMLP is NP-complete.

Proof. We first show that decision OMLP problem is in NP. If one correctly guesses the set of roles $r_1, r_2..r_j$, it only takes polynomial time to compute

$$\parallel r_1 \bigcup r_2 \bigcup ... \bigcup r_j - Q \parallel_1$$

and ensure that it is less or equal to θ, and that the number of roles j is not more than k where the weight of these redundancy permissions is not more than ω. Therefore, the decision OMLP problem is in NP.

To prove the NP-hard, we consider a subclass of the problem-the basic minimizing-approx principle of least privilege problem. According to the Theorem 3, we come to the conclusion that the basic minimizing-approx principle of least privilege decision problem is NP-hard. Hence the decision OMLP is NP-complete. □

5. Algorithms

In RBAC systems, the principle of least privilege should be enforced. In other words, if there is a set of requested permissions, we need select a set of roles to satisfy the basic principle of least privilege or its variants. The problems are all well known to NP-complete, and hence it is almost impossible to determine the solutions for any large sized problems in the polynomial time. Although paper [5] has proposed an algorithm to solve the basic principle of least privilege problem, however, it did't consider the weight of each permission and role, and furthermore, the algorithm cannot solve the δ-approx principle of least privilege problem and the minimizing-approx principle of least privilege problem. Hence, we express the basic principle of least privilege problem as an optimization problem: the Minimal Cost Set Covering problem (MCSC) [11]. We will show that the basic principle of least privilege problem is identical to the MCSC problem, then all results regarding the approximation of one of these problems carry over to the other. We can directly use the algorithms for the MCSC problem to solve the basic principle of least privilege problem. And then we design the alternative algorithms to solve the δ-approx principle of least privilege problem and the minimizing-approx principle of least privilege problem.

5.1. Mapping the Basic Principle of Least Privilege problem to the Minimal Cost Set Covering Problem

We first give a mapping way to solve the basic principle of least privilege problem, which is based on the idea of Theorem 4.

Definition 26. (MCSC Problem) Given a universe $\mathcal{U}$, a collection $\mathcal{C}$ of subsets of $\mathcal{U}$ whose union is $\mathcal{U}$, and each of the given subset has an associated cost ω from a weight function

Table IV. A sample for the minimal cost set covering problem

	e_1	e_2	e_3	e_4	e_5	e_6	
$\mathcal{C}_1$	1	1	0	0	1	1	ω_1
$\mathcal{C}_2$	0	1	1	0	1	1	ω_2
$\mathcal{C}_3$	1	1	0	1	1	0	ω_3
$\mathcal{C}_4$	0	0	0	1	1	0	ω_4

$\omega : \mathcal{C} \longrightarrow \mathbb{R}^+$, find a subset $\mathcal{D} \subseteq \mathcal{C}$ such that

$$\mathcal{U} = \bigcup_{D \in \mathcal{D}} D \text{ and } \sum_{D \in \mathcal{D}} \omega(D) \text{ is minimized}$$

We consider the MCSC problem is a binary matrix of size $m \times n$ where the number of rows, m, can be viewed as the number of collection $\mathcal{C}$ of subsets of $\mathcal{U}$ and the number of columns, n, can be viewed as the number of elements in $\mathcal{U}$. A 1 in cell $\{i, j\}$ $(i = 1, ...m; j = 1, ...n)$ denotes that subset i has the element j. Each row of the binary matrix will be assigned a positive weight ω_i $(i = 1, ...m)$. The goal is to choose a set of rows of minimal weight such that for all columns in the matrix, there is a 1 in at least one of the rows corresponding to the rows chosen. We present a sample in Table IV where e_i $(i = 1, 2, 3, 4, 5, 6)$ describes the elements in $\mathcal{U}$, $\mathcal{C}_i$ $(i = 1, 2, 3, 4)$ describes the subsets of $\mathcal{U}$ and ω_i $(i = 1, 2, 3, 4)$ describes the weight of each subset respectively. In this example, the goal is to choose a set of $\mathcal{C}_i$ $(i = 1, 2, 3, 4)$ of minimal cost such that all the elements e_i $(i = 1, 2, 3, 4, 5, 6)$ can be included into the selected subsets of $\mathcal{U}$.

Formally, we can reduce the basic principle of least privilege problem to the MCSC problem as follows.

Theorem 4. The basic principle of least privilege problem is identical to the MCSC problem when each subset has the same cost.

Proof. To show that the two problems are identical we show that their inputs and outputs exactly match. Thus, for every input instance, the output of both problems has a direct one-to-one mapping [14, 15].

- The input to both problem is a $m \times n$ boolean matrix;
- For any problem instance, the MCSC problem returns a set of rows that completely covers all the columns while minimizing the number of rows (Here, each row has the same cost). Each row corresponds to a role, R. For each row, we extract the set of columns C, in the row. For each column $c \in C$, add the assignment $\{c, R\}$ to PA. Similarly, for each row, assign the cost ω as the weight of each role, and set all the columns as the requested permissions set RQ;
- The resulting set of roles when each role has the same weight is guaranteed to be a solution to the basic principle of least privilege for the requested permissions set RQ. (i.e., the resulting set of roles are 0-consistent with the requested permissions set RQ, and the number of resulting roles is minimal). To prove the 0-consistency, it is sufficient

to note that all the resulting roles and the requested permissions set RQ gives us the original input matrix where the union of the resulting roles is equivalent to the original requested permissions set RQ. We can prove the minimality by contradiction. Assume that a different solution to the basic principle of least privilege problem exists-consisting of R' where $|R'| < |R|$. In this case, we can transform this solution into a corresponding solution for the MCSC problem. For each role $r' \in R'$, create the corresponding subset $C_{r'}$ consisting of the permissions given by PA. The union of all subsets $\bigcup_{r' \in R'} C_{r'}$ is 0-consistent with the requested permissions set RQ. However, the number of subsets is the same as $|R'|$ which is less than $|R|$. But that means the earlier solution is not minimal-and we have a contradiction. Therefore, the solution to the MCSC problem directly maps to a solution for the basic principle of least privilege problem. □

Thus the MCSC problem exactly corresponds to the basic principle of least privilege problem when each subset has the same cost. Then we can directly use the algorithms for the MCSC problem to solve the problem of the basic principle of least privilege. The detailed analysis of the algorithm can be found in [16, 17]. We now briefly present the genetic algorithm for the basic principle of least privilege. It consists of six steps as follows.

- **Representation:** Use an n-bit string (one bit per role) to represent a particular choice of roles. A value of 1 in the xth (where $x = 1, ...n$) position in string implies that role x is chosen to be in the cover;
- **Start:** Select a population of m points $x_1, ...x_m$ to represent the roles set at random and evaluate the roles set using evaluation function. Set the number of trial $t \leftarrow 0$;
- **Crossover/Mutate:** Pick two points x_i, x_j from the population using a probability distribution over their values. Select an integer from 0 to m at random with probability P_{cross}, and create two offsprings roles x_c and x_d using crossover; For each bit in x_c and x_d, with probability P_{muta}, alter its value;
- **Reproduce:** Pick two points x_e and x_f from the population using a probability distribution over the inverse of their values. Replace x_e and x_f in the population with x_c and x_d;
- **New Generation:** Repeat the above two steps until n points have been replaced in the population for the current t;
- **Stopping Criteria:** Set $t \leftarrow t + 1$. If the number of trial is larger than the maximum number of trial, or, for some point x_i, the value of evaluation is invariant, stop. Otherwise go to step 3.

The genetic algorithm consists of two phases. In the first phase, we select a population of roles set at random and evaluate those roles set. Hence we can find the better populations that have good fitness for the crossover or mutation. In the second phase, we generate the new generations based on crossover or mutation with a probability distribution. The algorithm will be stopped until the number of trial is larger than the maximum number of trial, or, the value of evaluation of the populations is invariant for some results. When the algorithm stopped we can get the role set that satisfies the basic principle of least privilege for the requested permissions set.

5.2. Algorithms for the Principle of Least Privilege Problems

In this section, we design the description methods and evaluation functions to deal with all the principles of least privilege problems using genetic algorithm.

- **The algorithm for the basic principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $sr + sp$ (where sr is the number of roles chosen in the cover, and sp is the penalty for cases where the chosen roles do not cover all the requested permissions set);
- **The algorithm for the optimal basic principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $m \times sr + wsr + sp$ (where m is the number of all roles, sr is the number of roles chosen in the cover, wsr is the weight of roles chosen in the cover, and sp is the penalty for cases where the chosen roles do not cover all the requested permissions set);
- **The algorithm for the δ-approx principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $sr + sp$ (where sr is the number of roles chosen in the cover, and sp is the penalty for cases where the chosen roles do not satisfy the δ-consistency with the requested permissions set);
- **The algorithm for the optimal δ-approx principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $m \times sr + wrp + sp$ (where m is the number of all roles, sr is the number of roles chosen in the cover, wrp is the weight of redundancy permissions, and sp is the penalty for cases where the chosen roles do not satisfy the δ-consistency with the requested permissions set);
- **The algorithm for the minimizing-approx principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $rp + sp$ (where rp is the number of redundancy permissions, and sp is the penalty for cases where the chosen roles do not satisfy the minimizing-approx principle of least privilege with the requested permissions set);
- **The algorithm for the optimal minimizing-approx principle of least privilege problem:** We use n-bit string (one bit per role) to represent a particular choice of roles. The evaluation function is defined as $n \times rp + wrp + sp$ (where n is the number of permissions, rp is the number of redundancy permissions, wrp is the weight of redundancy permissions, and sp is the penalty for cases where the chosen roles do not satisfy the minimizing-approx principle of least privilege with the requested permissions set).

6. Experimental Results

To show the advantage of our algorithms, we will present the most relevant results by implementing them on an Intel (R) Core (TM)D 2.2G machine with 2GB memory to evaluate how well our algorithms perform using different metrics. The running platform is Windows XP. We choose the parameters $p_{cross} = 0.75$, $p_{muta} = 0.25$.

To study the performance of our algorithms, we generate the synthetic test data as follows. First, the maximum number of roles, permissions, and each permission's weight are created respectively. Then we use *for* loop to create the relationships between roles and permissions. For each role, a random number of permissions are chosen. The value of each element in the matrix is randomly chosen as 0, indicating that the role has no such permission, or 1, indicating that the role has such permission. Finally, we preprocess the generated data, compute each role's weight, and select the union of all roles' permissions to construct the requested permissions set RQ.

We present the evaluation of our algorithms on two different role-permission assignments. For the first set of assignments, we fix the number of roles, while changing the number of permissions. Table V shows the test parameters. Each test is repeated one hundred times to evaluate the performance of the algorithms. We are interested in two things: the average number of evaluations taken by each algorithm to find the best solution in different situations and how well to find them. The average number of evaluations found by the OBLP algorithm for various length of requested permissions set is shown in Figure 1(a) (Here we only consider the OBLP algorithm because the BLP algorithm is simply a special case of the OBLP algorithm when each role has the same weight). The figure shows that when the length of the requested permission set is low, there is a low number of evaluations to find the best solution. As the length of the requested permissions set increases, the number of evaluations will increase accordingly. We can also see that our algorithm exhibits a polynomial increase in the number of solution evaluations to get to the best solution. This is contrast to the exponential increase in complexity of the only known algorithm guaranteed to find the optimal solution, namely exhaustive search. Thus, the algorithm is able to generate quality solutions in a very reasonable time frame.

In order to verify the accuracy of the solutions obtained, we implemented an exhaustive search algorithm to give us the solution for reasonable size problems. We denote all the results found by the exhaustive search algorithm as $S = \{rs_1, rs_2...rs_n\}$. When each test, the algorithm can get one role set rs. We let $k \leftarrow k + 1$ if rs exactly matchs to one of the role set rs_j $(1 \leq j \leq n)$ in S. Hence, the accuracy of the algorithm can be defined as follows.

Definition 27.(The accuracy of the Algorithm) The accuracy of the algorithm is measured as the ratio of the number of k to the algorithm running times.

The exactly matching can be defined as follows.

Definition 28. Given one roles set *ors*, the generated roles set *drs* is said to exactly match the original roles set *ors* if and only if each role in *ors* is also in *drs* and vice verse.

Figure 1(b) shows the average accuracy of the solutions that it finds in different length of requested permissions set. From the figure, we can see that, when the length of the requested permissions set is low, there is a high accuracy. As the length of the requested permissions set increases, the accuracy of the solution will decrease. This is because the algorithm is always able to generate optimal solutions to small size problems. For medium size problems the algorithm may or may not be able to produce optimal solutions. However, all of the accuracy is more than 90%, which means that our algorithm can generate quality solutions.

In the second experiment, we fix the number of permissions, while changing the number of roles. Table VI shows the test parameters. Figure 2(a) shows the average number of evaluations under different data set. Figure 2(b) shows the average accuracy under different data set. It

Table V. Parameter settings for testing performance (Fixed roles, varying permissions)

Name	*No. of roles*	*No. of permissions*
data1	100	200
data2	100	400
data3	100	800
data4	100	1600

Table VI. Parameter settings for testing performance (Fixed permissions, varying roles)

Name	*No. of roles*	*No. of permissions*
data1	40	800
data2	80	800
data3	160	800
data4	200	800

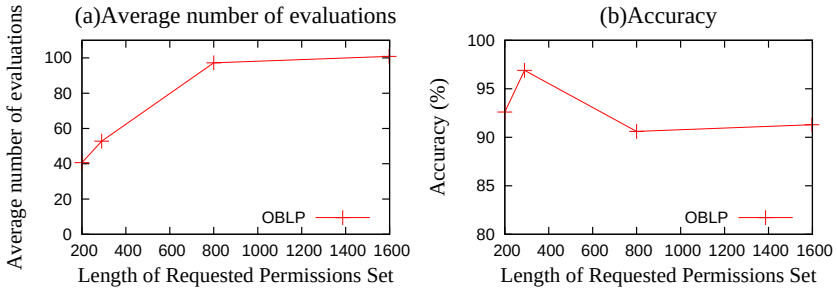


Figure 1. Performance comparison under the fixed number of roles for the OBLP algorithm

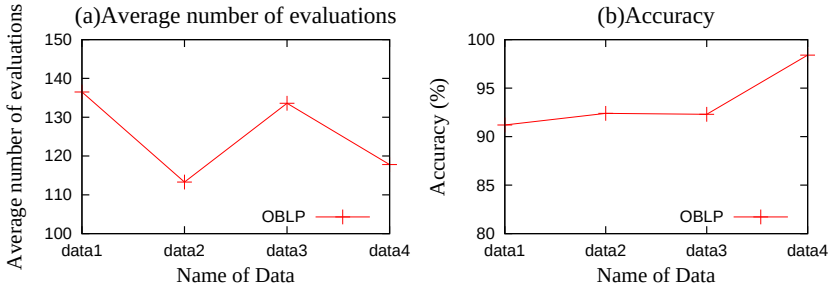


Figure 2. Performance comparison under the fixed number of permissions for the OBLP algorithm

can be clearly seen that the average number of evaluations is close to 120 when we look at the largest size data, and close to 140 when we look at the smallest size data. This is reasonable because the algorithm depends on the generated data but not on the size of the data when the length of the requested permissions set is the same. We can also see that our algorithm is much more preciser.

Figure 3(a) shows the average number of evaluations found by the OδLP algorithm for various length of the maximum number of δ under the first set of experiments, while Figure 3(b) shows the accuracy of the algorithm (Here we only consider the OδLP algorithm because the δLP algorithm is simply a special case of the OδLP algorithm when each permission has the same weight). Here, we can see that the average number of evaluations actually increases with the maximum number of δ in the same data on one hand, on the other hand the average number of evaluations increases with the size of the data. It can also be clearly seen that the accuracy is close to 100% in the smallest data (100 roles and 200 permissions), and close to 95% in the largest data (of 1600 permissions). This is quite good.

Figure 4(a) shows the average number of evaluations found by the OδLP algorithm for various length of the maximum number of δ under the second set of experiments, while Figure 4(b) shows the accuracy of the algorithm. Here, we can see that the average number of evaluations actually increases with the maximum number of δ. Again, the accuracy of the algorithm is quite good, with the largest data and largest number of δ getting approximately

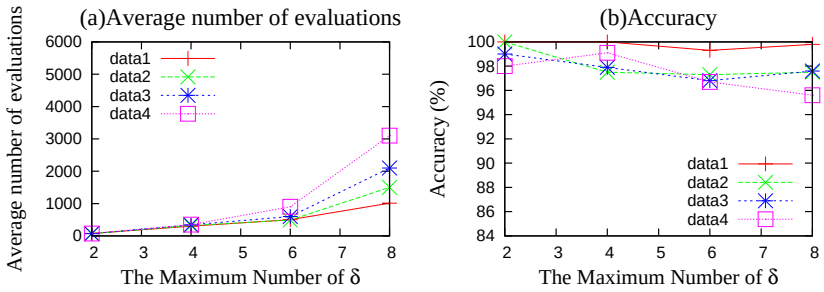


Figure 3. Performance comparison under the fixed number of roles for the OδLP algorithm

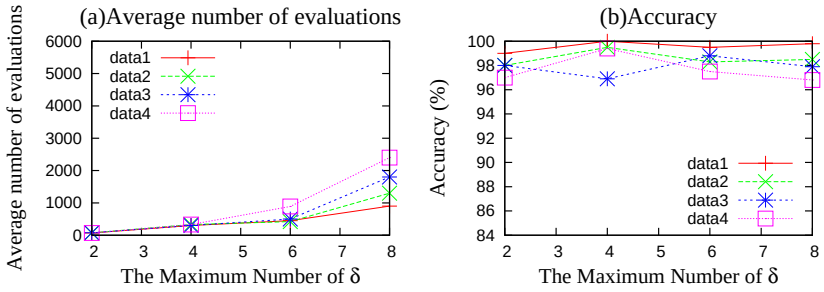


Figure 4. Performance comparison under the fixed number of permissions for the OδLP algorithm

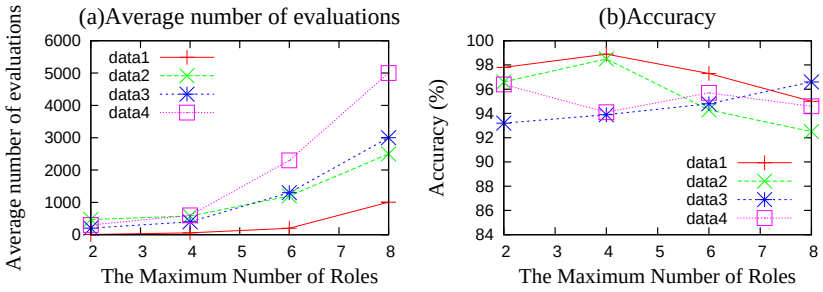


Figure 5. Performance comparison under the fixed number of roles for the OMLP algorithm

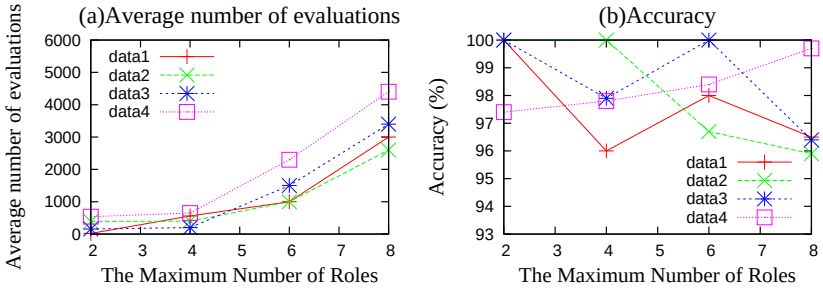


Figure 6. Performance comparison under the fixed number of permissions for the OMLP algorithm

to 97%. Figure 5(a) shows the average number of evaluations found by the OMLP algorithm for various length of the maximum number of roles under the first set of experiments, while Figure 5(b) shows the accuracy of the algorithm (Here we only consider the OMLP algorithm because the MLP algorithm is simply a special case of the OMLP algorithm when each permission has the same weight). Figure 6(a) shows the average number of evaluations found by the OMLP algorithm for various length of the maximum number of roles under the second set of experiments, while Figure 6(b) shows the accuracy of the algorithm. From these figures, we can see that, our algorithm can also find the best solution taking lesser evaluations.

7. Conclusions

While exact match is a good thing to have, approximate match might be a prudent choice for the real systems. To solve this problem, this paper addresses the problem of the δ-approx principle of least privilege, the minimizing-approx principle of least privilege. However, how to choose from the different sets which can enforce the same principle of least privilege for the same requested permission set? To solve this problem, this paper introduces the weight of permission and role set. It has been proved that the basic principle of least privilege and all its variants are NP-Complete. Therefore, we first reduce the basic principle of least privilege

to the MCSC problem. As a result, we could borrow the implementation solutions proposed for the MCSC problem and directly apply them to solve the problem of the basic principle of least privilege. We also use alternative algorithms to solve its variants. Finally, we carry out the experiments to illustrate the effectiveness of the proposed techniques. As has been proved, the proposed approach has superior performance in finding the optimal solution to improve both speed and accuracy. In our future work, we will try to find more significant attributes to make the weight of permission and role more accurate and meaningful. So that they could be used to further optimize the results.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China under Grant 60873225, 60773191, 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, Natural Science Foundation of Hubei Province under Grant 2009CDB298, Wuhan Youth Science and Technology Chenguang Program under Grant 200950431171, Open Foundation of State Key Laboratory of Software Engineering under Grant SKLSE20080718, Innovation Fund of Huazhong University of Science and Technology under Grant Q2009021.

REFERENCES

1. Sandhu R, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *IEEE Computer* 1996; **2**9(2):38–47.
2. D.Richard Kuhn. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. *Proceedings of the Second ACM Workshop on Role-based Access Control.* ACM Press: New York, NY,U.S.A., 1997; **2**3–30.
3. David F.Ferraiolo, Janet A.Cugini, D.Richard Kuhn. Role-based access control (RBAC): features and motivations. *Proceedings of the 11th Annual Computer Security Applications Conference.* IEEE Computer Society Press, 1995; **2**41–248.
4. J.Saltzer, M.Schroeder. The protection of information in computer systems. *Proceeding of the IEEE* 1975; **63**(9):1278–1308.
5. Liang Chen, Jason Crampton. Inter-domain role mapping and least privilege. *Proceedings of the 12 ACM Symposiums on Access Control Models and Technologies (SACMAT),*Sophia Antipolis, France, June 2007; **1**57–162.
6. Fred B. Schneider. Least privilege and more. *IEEE Security and Privacy.* IEEE Educational Activities Department: Piscataway,NJ,U.S.A.,2003; **5**5–59.
7. Ruixuan Li, Zhuo Tang, Zhengding Lu, Jinwei Hu. Request-Driven Role Mapping Framework for Secure Interoperation in Multi-Domain Environments. *International Journal of Computer Systems Science and Engineering* 2008; **23**(3): 193–207.
8. Jinwei Hu, Ruixuan Li, Zhengding Lu. Establishing RBAC-based secure interoperability in decentralized multi-domain environments. *Proceedings of the 10th International Conference on Information Security and Cryptology,*Seoul, Korea, 2007; **4**9–63.
9. Ninghui Li, Mahesh V.Tripunitara, Ziad Bizri. On mutually-exclusive roles and separation of duty. *ACM Transactions on Information and System Security.*ACM Press: New York, NY,U.S.A., 2005; **10**(2): 1–36.
10. Xinyu Wang, Jianling Sun, Chao Huang, Di wu. Security violation detection for RBAC based interoperation in distribution environment. *IEICE Trans.Inf.and Syst.* May, 2008; **E91-D**(5):1447–1456.
11. T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein. *Introduction to algorithm.* (2nd edn). The MIT Press, 2001.
12. Xiaopu Ma, Ruixuan Li, Zhengding Lu. Role mining based on weights. *Proceedings of the eleventh ACM symposium on access control models and technologies,*Pittsburgh, Pennsylvania, USA, June 2010.
13. M.R.Garey and D.S.Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H.Freeman, New York, NY, USA, 1979.

14. Pauli Miettinen, Taneli Mielikainen, Aristides Gionis, Gautam Das, Heikki Mannila. The discrete basic problem. *IEEE Transcations on knowledeg and data engineering.*October, 2008; **20**(10): 1348–1362.
15. Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo. The role mining prolbem: finding a minimal description set of roles. *Proceedings of the 12 ACM Symposiums on Access Control Models and Technologies (SACMAT),*Sophia Antipolis, France, June 2007; **1**75–184.
16. J.E.Beasley, P.C.Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research* 1996; **94**(2):392–404.
17. Darwin Gouwanda, S.G.Ponnambalam. Evolutionary search techniques to solve set covering problems. *World Academy of Science, Engineering and Technology* 2008; **39**:20–25.