Dynamic User-Oriented Role Based Access Control Model (DUO-RBAC)

Hazem Kiwan [*] and Rashid Jayousi [**]

[*] Department of Computer Science, Al-Quds University
roone.hazem@gmail.com
[**] Department of Computer Science, Al-Quds University
rjayousi@staff.alquds.edu

Abstract. Most researchers use role mining to generate role-based access control model from the existing user-permission assignments. User-oriented role-based access control model is a type of role-based access control model, which aims to use role mining from an end-user perspective to generate an RBAC model. This research is the first for generating a dynamic user-oriented role-based access control model for inserting a new user-permission assignments to the existing model re-generating roles, with a constraint that there are no changes in the number of role assignments for each user in the system after the insertion process, since the user will be conflicted if he has different number of roles from time to time. Also, we have developed a new algorithm, which based on user-oriented role mining to find the way to insert the new UPA to the existing model. Our experiments applied on benchmark "Access Control" real datasets to evaluate the results.

1 Introduction

Role Based Access Control (RBAC) is a type of access control models (permission models) that allows authorized users to do their tasks and perform their actions while they are browsing their system. The permissions of users in the RBAC model are fetched through role mining to generate roles based on existing user permission assignments. Then, the permissions are assigned to corresponding roles and roles are mapped to the users of the system.

The key idea of role mining is to utilize the data mining technologies to discover a good role set of permissions which are existing in user-permission assignments of the old access control system. The discovered roles are then applied to the system through the activities of the corresponding users (Lu et al. (2015)). In short, the existing studies investigate the concept of role mining in different objectives, such as minimization the administrative cost, minimization of the number of roles, minimization of the complexity of the role hierarchy structure introduced by Molloy et al. (2008), security administration like (Kuhlmann et al. (2003)), and user-oriented role mining.

There are different algorithms that use role mining to generate user-oriented role-based access control model (UO-RBAC) from existing user-permission assignments in the system.

But there aren't any dynamic algorithms that take into account whether we have new users with new permissions (new user-permission assignments UPA) that join the system after the model is generated (Don't have a dynamic model which depends on a dynamic algorithm to insert those new UPA to the generated model in the system). However, the only two ways to insert the new user-permission assignments UPA to the generated model in the system without a dynamic algorithm are:

1. Add the new UPA to the original dataset which contains the existing user-permission assignments before generating the model, then rerun the algorithm to the dataset (after adding the new UPA) to generate a new optimal user-oriented RBAC model. The problem of this solution is when the model is regenerated, we would have a new user-role assignments for each user and a different number of role assignments for some users, in this case, the user will be conflicted when the number of his/her role assignments change from time to time. So, the main concept of generating the model from perspective will not be available.
2. Assign all the permissions for each new user to a new role, then assign this role for the user that contains his/her permissions (each user from the new users will be assigned to one role which contains his/her permissions). The problem with this solution is that the total number of roles and the total number of user-role assignments are not optimal. So, in this case, we do not have an optimal user-oriented RBAC model.

In this work we aim to build a dynamic user-oriented RBAC model to insert the new users with their permissions (new UPA) to the existing user-oriented RBAC model by a dynamic algorithm with Making sure that after the insertion process we will still have the following:

— A user-oriented role-based access control model.

— There are no changes in the number of role assignments for each user in the system after the insertion process, since the user will be conflicted if he has different number of roles from time to time.

This research is conducted through the following: Chapter 2 a quick through about previous works on role engineering, role mining, role-based access control, and user-oriented role-based access control. Chapter 3 describes our proposed dynamic user-oriented RBAC model (DUO-RBAC) and dynamic user-oriented RMP algorithm. Chapter 4 experiments on access control data sets and analysis of results. Chapter 5 concludes the work.

2 Related Work

"Role Engineering is a security-critical task for systems using role-based access control (RBAC)" (Frank et al. (2008)). Coyne (1996) introduced role engineering concept to find a correct and a complete architecture structure to generate a business function and organization's security policies. They used a top-down process-oriented strategy to produce and generate roles. They also introduced role engineering which is used for role-based access control as the process of defining roles, permissions, constraints, and role-hierarchies (Frank et al. (2008)). In this research, the authors applied a novel scenario-driven role engineering process to introduce and generate roles in different case studies. The bottom-up approach for role engineering was used in (Ene et al. (2008)) to find a set of roles (R), with a set of user-role assignments $(U)A$ and role-permission assignments (PA).

Le et al. (2012), introduced the concept of role mining to enhance one of the existing access control models. The authors use an existing algorithm and apply it to the RBAC model. This algorithm depends on the data mining concept. It also studies the behavior of the end-user to determine the perfect algorithm they used. However, Ene et al. (2008), the authors used a Lattice algorithm to minimize the number of the roles that each user has with the same permissions. The authors formulate several role engineering problems that introduced by Vaidya et al. (2007) to develop an optimal algorithm. Ma et al. (2010), added a weight to the role mining process. The focus of their research was given a weight for permissions to reflect their importance to the system, they assigned each permission to a weight in feasible ways. They also used a matrix to present the relationships between the user and the permissions within the system, then they calculated the similarities and found out how to define the weight for permissions based on this similarities. There are two limitations to these researches. The first limitation is to generate a role-based access control model by using role mining for the existing user-permission assignments without taking into account if there are new users with new permission needed to insert to the system after generating the model. Second, they focused on generating the model with a minimum number of role and role-permission assignments to reduce the complexity of role hierarchy and reduce the administrative cost on the system. On the other hand, our study focuses on implementing a new dynamic role-based access control model based on the user-oriented concept and role mining concept. Also, this research is using a new dynamic algorithm to insert the new users with their new permissions to the system after generating the model.

The concept of user-oriented role-based access control was introduced by Kuhlmann et al. (2003). This research was the first and only research (as they mentioned) that used user-oriented role mining to define role mining from a user's perspective. Their study depends on generating a user-oriented RBAC model by assigning each user in the system to as few as possible roles; since the user does not prefer to being overwhelmed by assuming too many roles. In the fact, each user would wish to have only one role assign to him/her, and it provides the all necessary access privileges related to his/her work and function smoothly. Actually, the most organization's systems had been designed that way. For example, in a healthcare system, each employee carry only one role, either MANAGER, ACCOUNTANT, PATIENT or DOCTOR. So, user-oriented role mining is characterized by the fact that the maximum role assignments for each user (defined as t) should be constrained. The authors used the concept of role mining to sparse user-role assignments, then they developed a user-oriented exact role mining problem (RMP) algorithm to generate a user-oriented role-based access control model from the existing user-permission assignments UPA in the system. They defined constraints to their algorithm while generating the model such as: using user-oriented role mining problem to finding the minimum number of roles from the candidate roles, completely reconstruct the existing user-permission assignments, and no user can have more than t roles (t defined at the beginning of their algorithm). Their experiment is conducted on a benchmark access control datasets. Then, the generated model contains total number of roles $|R|$, total number of role-permission assignments $|PA|$, and the total number of user-role assignments $|UA|$ for each dataset, the number of direct user-permission assignments $|UPA|$, and the number of edges in the reduced role hierarchy $|t_reduce(RH)|$. Those five main factors can be used to evaluate the feasibility of an RBAC model as mentioned in (Neumann and Strembeck (2002)). But in the case of user-oriented RBAC model, they used the first three factors to evaluate their model since no further exposition is needed on them. Also, they applied a weighted structural complexity measure introduced in (Neumann and Strembeck (2002)) added extra evaluative criteria to evaluate the model.

$$\begin{aligned}&\min w_r * |R| + \lambda|UA| - \lambda * t\\ &s.t.\{UPA_{mxn} = UA_{mxk} \otimes PA_{kxn}\end{aligned} \tag{1}$$

This objective function was developed by Kuhlmann et al. (2003) depends on using user-oriented exact role mining problem (RMP). It can be affected by a number of roles, a number of user-role assignments and the weight for each role in the model. We must get the optimal number of roles and user-role assignments to make the objective function minimized. Now, the question here is 'how did they get this objective function?'. Suppose we have n permission, m users, user-permission assignments UPA_{mxn}, and positive number t, and we will use all these data to discover user role assignments and role set PA_{kxn} and user-role assignment UA_{mxk} under constraints that total number of roles k minimized, user-role assignments and role-permission assignments completely reconstruct the existing user-permission assignments, and no user has more than t roles. Kuhlmann et al. (2003) Described those constraints mathematically by the following functions:

$$\begin{aligned}&\min k\\ s.t.&\begin{cases}UA_{mxk} \otimes PA_{kxn} = UPA_{mxn}\\ \sum_j UA(i,j) \leq t, \forall i\\ UA \in \{0,1\}^{mxk}, PA \in \{0,1\}^{kxn}\end{cases}\end{aligned} \tag{2}$$

$$\begin{aligned}&\min |R| + \sum_i \lambda_i(\sum_j UA(i,j) - t)\\ &s.t.\{UPA_{mxn} = UA_{mxk} \otimes PA_{kxn}\end{aligned} \tag{3}$$

Where is λ_i the Lagrange multiplier for the constraint of $\sum_j UA(i,j) \leq t$

Further, we could assume that all Lagrange multipliers have the same value of λ (Kuhlmann et al. (2003)). Then we will have the objective function as shown in the $equation 1$. More details in (Kuhlmann et al. (2003)). However, the only one limitation in this research is withered we need to add new users with their permissions (new user-permission assignments) to the system which already has a generated user-oriented RBAC model. In this research, there are only two existing solutions. Compare to our approach, our solution is dynamic user-oriented role-based access control. We developed a dynamic algorithm depends on user-oriented role mining to add these new user-permission assignments to the existing model without using any of the two previous solutions. In this case, we will make sure that the total number of roles and the total number of user-role assignments in the model after the insertion process minimized, no user have roles more than t that predefined in the existing model, and we also keep the RBAC model suitable for user's perspective.

3 DUO-RBAC Model

This chapter describes our proposed dynamic user-oriented role-based access control model (DUO-RBAC) and dynamic user-oriented RMP algorithm. DUO-RBAC is a model for inserting new users with their permissions to the system which already has a generated user-oriented RBAC model. Our model uses dynamic RMP algorithm to insert those new users with their permissions to the system.

3.1 DUO-RBAC Model Overview

The first step for or generating a dynamic user-oriented role-based access control model is having an existing generated user-oriented RBAC model and a list of new users with their permissions UPA that needs to be added to the model. Then, those two sections will enter into a dynamic user-oriented RMP algorithm. Finally, a new user-oriented RBAC model will be generated which contains the new users and their permissions.

After the model is generated, each user in the system is assigned a number of roles that don't exceed t (Lu et al. (2015)). Also, each role is assigned to one or more permission in the system. The result is that the new users with their permissions are in the system without changing any existing assignments in the existing model before running our model. Through this method, each old user in the system still has the same role assignments and each new user joining the system has new assignments to an existing role or to a new role which is generated through the insertion process.

3.2 Dynamic User-Oriented RMP Overview

The algorithm that we use in DUO-RBAC Model is called dynamic user-oriented RMP. Our dynamic algorithm is the first algorithm to insert new users with their permissions to an existing user-oriented RBAC model. It introduces an optimal way to insert all users and their permissions to the existing user-oriented RBAC model. It also covers all possible cases when we add those new users. Our dynamic algorithm depends on the concept of dynamic role generation (Vaidya et al. (2006)). It focuses on the new UPA dataset (users with their permissions) and the existing user-oriented RBAC model. It inserts user one by one to the system with an optimal way to keep the increasing number of roles and user-role assignments UA perfectly. It also makes sure that the constraint of users not being assigned roles that doesn't exceed t is achieved (Lu et al. (2015)). The dynamic user-oriented RMP, is an algorithm to insert a new UPA dataset to the existing user-oriented RBAC model that is generated from running user-oriented exact RMP algorithm on the old dataset, and finding the optimal final number of roles and user-role assignments the constraint of not having users assigned to roles that don't exceed t (Lu et al. (2015)).

3.3 Dynamic User-Oriented RMP Structure

In the beginning, and before describing our algorithm, we apply a preprocessing step to remove the all users that have the same permissions, then replace them by one user to reduce the size of the new dataset that will enter the model. This step is applied to other role mining algorithm, such as (Molloy et al. (2010)). The following example shows how to apply the first preprocessing step on the new dataset.

Algorithm 1: Preprocessing step algorithm

```
Input: $UPA$
Output: $UPA'$
if $\forall UPA_i \in UPA\, s.t.\, UPA_i \notin UPA'$ then
    UPDATE $UPA'$ by adding $UPA_i$;
end
```

	p1	p2	p3	p4	p5	p6
u1	1	0	1	0	1	1
u2	1	1	0	0	0	0
u3	1	1	0	1	0	0
u4	1	1	1	0	1	0
u5	1	1	0	1	0	0
u6	1	0	0	0	0	1
u7	0	1	0	0	1	1
u8	1	0	1	0	1	1

TAB. 1: Existing (UPA)

	p1	p2	p3	p4	p5	p6
u2	1	1	0	0	0	0
u4	1	1	1	0	1	0
u5	1	1	0	1	0	0
u6	1	0	0	0	0	1
u7	0	1	0	0	1	1
u8	1	0	1	0	1	1

TAB. 2: Remaining (UPA)

If we take a look on $table1$, we will note that $u1$ has the same permissions that $u8$ has and $u3$ has the same permissions that $u5$ has. So, in this state, we will replace the two users in each case by one user, in this way we reduced the size of the dataset by deleting two users. $table2$ shown the remaining UPA.

The structure of our dynamic algorithm depends on entering the new users one by one to the model, then finding the optimal way to assign this user to an existing role or generate a new role then assigns this user to it. In the process of our algorithm we are making sure that all user-role assignments for each user in the existing user-oriented RBAC model do not change, have an optimal final number of roles (R), user-roles assignments (UA), and achieve the constraint that there aren't any users assigned roles that don't exceed t (Lu et al. (2015)). In our algorithm, the new users with their permissions are entering one by one. The inputs of our algorithm are: UPA, UPA_i, t, UA and PA. And the outputs of our algorithm are: UA and PA.

The first step of our algorithm is to compare user's permissions with role-permission assignments and check if the user has a permission that is not covered by any role. In this case, the algorithm creates a role which contains those all permissions and assigns this role to a user, this step also applied in (Lu et al. (2015)). If all user's permissions are covered by roles, then our algorithm creates a combination of candidate roles. It focuses on the existing role-permission assignments and checks if there is a role assigned to permissions and the new user partially or entirely has those permissions (role is not assigned to any other permissions), then add these role-permission assignments to a combination as a candidate role.

Algorithm 2: Dynamic user-oriented RMP algorithm

```
Input: $UPA, UPA_i, PA, UA, t$
Output: $PA, UA$
if $\exists p' \in UPA_i \; s.t. \; p' \notin (P \; in \; PA)$ then
    $\forall p' \in UPA_i, \; r \leftarrow \{r, p\}$;
    UPDATE $PA$ by adding $r$;
    UPDATE $UA$;
end
else
    $\forall PA_j \; s.t. \; PA_j / UPA_i = \phi, \; CRoles \leftarrow \{CRoles, PA_j\}$;
    $\forall \, CRoleSet = \{CRoles_1 \; TO \; CRoles_t\}$
      $s.t. \; CRoles_1 \cap CRoles_2 \cap .. \cap CRoles_t = \phi$,
      $Combinations \leftarrow \{Combinations, CRoleSet\}$;
    if $\exists \, Combinations_i \; s.t. \; UPA_i / Combinations_i = \phi$
      $AND \; |Combinations_i| < t$ then
        UPDATE $UA$ by adding $Combinations_i$;
    end
    else
        $\forall UPA_i \in UPA, \; s.t. \; UPA_i / Combinations_i$;
        $CRoles \leftarrow \{Combinations_i, UPA\} \; in \; UPA_i / Combinations_i$
          $s.t. \; |P| \; is \; min \; AND \; |CRoles| < t$;
        UPDATE $PA$ by adding $r$;
        UPDATE $UA$ by adding $CRoles$;
    end
end
```

After that, we will have a combination of candidate roles which the new user may be assigned. After we have a combination of candidate roles, the algorithm checks if we can create a set of candidate roles from the combination. In this case, we have two constraints: The number of candidate roles that doesn't exceed t, and those candidate roles cover completely all permissions for the new users. If those two constraints are applied, the algorithm assigns the new user to this candidate role without creating any new roles for the new user. The last step is applied if none of the previous steps cover the current user. This step is applied if one of the following two cases is valid. First, if we cannot find a set of candidate roles from the combination to completely cover all new user's permissions (the user still has uncovered permissions). Second, if the number of candidate roles in the set that completely cover all new user's permissions that don't exceed t. In these two cases, the algorithm creates a new role to include the uncovered permissions, the user should be assigned to some of the candidate roles in the set, but the question here: What candidate roles the algorithm will choose? To answer this question, first, we studied some of the strategies that are used to choose a candidate role such as, (Lu et al. (2015)), (Ene et al. (2008)), and (Molloy et al. (2010)). After we have tried each strategy separately on our algorithm, we found that all of them do not work well in our case. So, we used an alternative strategy: we choose to make the selection of candidate roles dynamically, it depends on what uncovered permissions will be left to utilize the created role which includes these left permissions on the assignments for the remaining users. Using this method, the new user is guaranteed to have all the uncovered permissions, and we reduce the amount of role creation in the system. Finally, we will assign the user to the created roles and to the candidate roles that have been chosen from the list.

4 Experiments and Results

Our experiments applied on benchmark "Access Control" real datasets, those datasets are apj, $domino$, cas_small, $healthcare$, $firewall1$ and $firewall2$. All those datasets are collected by (Ene et al. (2008)). Table 3 shows the data description for each dataset that includes number of users, number of permissions and number of user-permission assignments.

Dataset	$\|U\|$	$\|P\|$	$\|UPA\|$
apj	2,044	1,164	6,841
domino	79	231	730
cas_small	3,477	1,587	105,205
healthcare	46	46	1,486
firewall1	365	709	31,951
firewall2	325	590	36,428

TAB. 3: Datasets Description

To validate the results after the experiments, we depend on an objective function. Because we are working on generating the model from an end-user perspective, the final RBAC model that generated after inserting new users with their permissions must be user-oriented.

Our experiment divided into three major steps. The first step we running user-oriented exact RMP on the existing real datasets to generate user-oriented RBAC model which contains roles, user-role assignments and role permission assignments as in (Lu et al. (2015)). After that, use these components to get the optimal results by using the objective function. The second step, we removed some users and their permissions from those datasets to have remaining data less than the original, then running user-oriented exact RMP on those remaining data to generate its user-oriented RBAC model and get the value from the objective function. The result of the objective function in this step should be optimal in case user-oriented, because we used user-oriented exact RMP, after that, we are now having a user-oriented RBAC model, and also we have new users and their permission (the removed users from datasets). The final step, inserting the new users (removed users) and their permissions to the existing user-oriented RBAC model to generate the final model. Note that, the number of roles, user-role assignments in the final model should be approximately the same as in the model that generated in the first step (before removing users), also, the roles assignments for each existing user in the model must not change (still the same as before inserting users).

Tables below show the results that we got after running user-oriented exact RMP algorithm on benchmark.

t	R	UA	$UA+PA$
2	23	79	716

TAB. 4: Domino

t	R	UA	$UA+PA$
2	18	46	545
3	18	53	468

TAB. 5: healthcare

t	R	UA	$UA+PA$
2	564	2044	5565
3	497	2218	5221
4	485	2277	5096

TAB. 6: apj

t	R	UA	$UA+PA$
2	90	365	7100
4	85	454	6890
6	84	600	6879
8	80	1516	6638

TAB. 7: fire1

After deleted a random number of users from each dataset and generated a user-oriented RBAC model without the deleted users by running user-oriented exact RMP algorithm. Tables below show the results that we got when running dynamic user-oriented RMP algorithm to insert the deleted to the existing user-oriented RBAC Model. Also, the tables show that the number of roles (R) inversely proportional with the value of t, since the minimum number of R was in a case that t is maximized, and the maximum number of R was in a case that t minimized. On the other hand, the number of user-role-assignments (UA) directly proportional with the value of t, since the minimum number of UA was in a case that is minimized and the maximum number of UA was in a case that t is maximized.

t	$D.U.$	R	UA	$UA+PA$
2	15	23	79	716
2	22	23	79	716

TAB. 8: Domino

t	$D.U.$	R	UA	$UA+PA$
2	10	18	46	545
3	10	16	53	481
2	14	18	46	545
3	14	15	58	490

TAB. 9: healthcare

t	$D.U.$	R	UA	$UA+PA$
2	50	563	2050	5596
3	50	495	2221	5240
4	50	480	2284	5149

TAB. 10: apj

t	$D.U.$	R	UA	$UA+PA$
2	25	90	365	7100
6	25	80	605	6936
2	36	90	366	7102
6	36	82	605	6938

TAB. 11: fire1

5 Conclusion

In this work, a dynamic user-oriented role-based access control model (DUO-RBAC) was designed, and a dynamic user-oriented RMP algorithm was developed. The DUO-RBAC is a complete model aimed to insert the new user-permission assignments (new UPA) to the existing model by using the dynamic algorithm to the existing model under two constraints which make our designed model more efficient than the existing ways. Also, we discussed the only two ways to insert the new user-permission assignments (UPA) to the generated model and introduced the limitations for each one in a different case. Compare to our work, the developed algorithm achieved the optimal total number of roles and total user-role assignments in the generated model after the insertion process. Our experiments based on using a benchmark access control datasets. To validate our returned results, we can make a comparison between them and the results in (Lu et al. (2015)). The number of user-role assignments in the dynamic model at each dataset was less than the original one, and the number of roles is equal to or more than the original one. By using this method, the evaluating function still has the same value in each case which is the optimal value in case user-oriented. Also, our experiment makes sure that each user in the system still has the same number of role assignments which achieve and keep the model in the system user-oriented.

References

Coyne, E. J. (1996). Role engineering. In *Proceedings of the first ACM Workshop on Role-based access control*, pp. 4. ACM.

Ene, A., W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan (2008). Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pp. 1–10. ACM.

Frank, M., D. Basin, and J. M. Buhmann (2008). A class of probabilistic models for role engineering. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 299–310. ACM.

Kuhlmann, M., D. Shohat, and G. Schimpf (2003). Role mining-revealing business roles for security administration using data mining technology. In *Proceedings of the eighth ACM symposium on Access control models and technologies*, pp. 179–186. ACM.

Le, X. H., T. Doll, M. Barbosu, A. Luque, and D. Wang (2012). An enhancement of the role-based access control model to facilitate information access management in context of team collaboration and workflow. *Journal of biomedical informatics 45*(6), 1084–1107.

Lu, H., Y. Hong, Y. Yang, L. Duan, and N. Badar (2015). Towards user-oriented rbac model. *Journal of Computer Security 23*(1), 107–129.

Ma, X., R. Li, and Z. Lu (2010). Role mining based on weights. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pp. 65–74. ACM.

Molloy, I., H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo (2008). Mining roles with semantic meanings. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pp. 21–30. ACM.

Molloy, I., H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo (2010). Mining roles with multiple objectives. *ACM Transactions on Information and System Security (TISSEC) 13*(4), 36.

Neumann, G. and M. Strembeck (2002). A scenario-driven role engineering process for functional rbac roles. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pp. 33–42. ACM.

Vaidya, J., V. Atluri, and Q. Guo (2007). The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pp. 175–184. ACM.

Vaidya, J., V. Atluri, and J. Warner (2006). Roleminer: mining roles using subset enumeration. In *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 144–153. ACM.