Object Based Dynamic Separation of Duty in RBAC

Muhammad Asif Habib and Christian Praher
FIM, Johannes Kepler University, Austria
habib@fim.uni-linz.ac.at, praher@fim.uni-linz.ac.at

Abstract

Role Based Access Control (RBAC) offers tight security of information and ease of management to implement. RBAC is a proven and open ended technology that is being attracted by most of the organizations for its capability to reduce security administration in terms of cost and complexity. The focus of this paper is one of the important factors in RBAC, i.e. Dynamic Separation of Duty (DSD) which is implemented to avoid internal security threats. We discuss DSD from a different perspective i.e. object based separation of duty. Different problems and observations have been described regarding DSD with respect to formal definitions of DSD. Those observations and problems influenced us to go for updated definition of DSD. So, we propose a newly updated definition of DSD. Different examples have been given regarding object based DSD with different scenarios. We also described benefits of implementing newly proposed definition of DSD.

1. Introduction

The success of a business depends on the ability and strength to protect the valuable information and data. Information and data are the most precious and valuable things for any organization. The organizations demand for fool proof security for their data and information as well as they demand for effective execution of business tasks after the implementation of a security policy. The organizations do not want the business processes to be disturbed after implementing any security policy. They do not want any such security policy that after implementing the policy, the organizations have to suffer for delay in execution of business processes or it slows down business tasks. So, the organizations demand for secure as well as efficient systems.

RBAC is known as the evolution in the field of Access Control. There are three main entities which are users, roles and permissions in RBAC. The users and permissions are assigned to roles. The permissions are comprised of operations and objects [2]. The basic story revolves around the roles. The concept of least privilege and separation of duty have given the opportunity to make the information and data more secure than before. According to ASCAA Principles for Next-Generation Role-Based Access Control [3], granting and revoking the roles will be performed dynamically in future. A detailed mechanism is given in [4] about the dynamic activation and deactivation of roles. So, dynamically granting and revoking roles can be implemented efficiently after the implementation of dynamic separation of duty in RBAC.

The definition of dynamic separation of duty has been given in ANSI Standard [1] as "A user can be authorized for multiple mutually exclusive roles and the user can exercise these roles independently but not at the same time or simultaneously". In this paper we propose a different definition of dynamic separation of duty. We consider object as another important entity like roles, users and permissions in RBAC. In this paper it is assumed that object can be any resource like file, folder, printer or directory etc. The importance of implementing dynamic separation of duty from object perspective is highlighted.

This paper is divided into different Sections. In Section 2, the overview of separation of duty and its types is discussed. In Section 3, the observation part which influenced us to go for updated definition of DSD is discussed. In Section 4, we discuss object based separation of duty, i.e. the core of this paper. In Section 5, a number of examples of different possibilities of role activation with respect to different scenarios are discussed. In last Section 6, we concluded the paper.

2. Overview

Role Based Access Control (RBAC) is getting fame day by day and is being implemented in organizations. In RBAC some of the important factors are the principle of least privilege and separation of duty [7, 10]. According to the concept of separation of duty, a business process or task is divided into more than one sub process or sub task. These sub tasks are assigned to different roles and different users are assigned to these roles. These roles are declared mutually exclusive to each other, i.e. these roles will not be activated by a single user

at the same time [1]. So there will be a least chance of fraud if a business task is involved with more than one user. Consequently one user would not be able to perform one complete business task independently.

The concept of separation of duty has a long history. It is used to extend the liability and obligation to more than one person to minimize the chance of fraud [7]. The separation of duty is a security parameter used to enforce security of an information system. The concept of Separation of Duty has been already discussed in different research papers [5, 6, 7, 11]. To minimize the chances of fraud, a business task is needed to divide into multiple sub tasks executed by more than one role. If a complete business task is executed by only one role then there are many more chances of fraud as compared to the situation where more than one role is involved in the execution of a business task. Different forms of separation of duty have been discussed in [6, 7, 8, 12] as Static Separation of Duty, Dynamic Separation of Duty, Object Based Separation of Duty, Operational Separation of Duty and History-based Separation of Duty.

Suppose if a single role has permissions to create, sign and finally approve the purchase order in a purchasing department then there will be a maximum chance of fraud, because only one person is involved in a complete business process. So nothing can stop a user who is authorized to have this role from exercising all the permissions and executing the whole business process which is against the concept of separation of duty. But if we break up this role in to different sub roles representing different business sub tasks and these sub tasks executed by different roles and all these roles are constrained not to attain by a single user then there will be least chance of fraud as compared to the previous situation. RBAC is a mechanism that is used to implement many policies but separation of duty is closely attached to RBAC because RBAC is considered as a natural mechanism for the implementation of separation of duty [13].

3. Observations

According to the definition given in [1] about dynamic separation of duty, one user can not activate more than one mutually exclusive role at the same time. If any user wants to activate another mutually exclusive role then the user has to quit its previously activated mutually exclusive role which means a user can have multiple mutually exclusive roles but not at the same time. We propose updated definition of dynamic separation of duty due to below observations that might be helpful for the proper implementation of dynamic separation of duty in the real spirit.

- If a user is assigned to two mutually exclusive roles and the user has activated one of those roles for one object then logically the user should be allowed to activate other mutually exclusive role for other object at the same time.
- It does not make sense if a user is not allowed to activate a mutually exclusive role for one object even though the user is authorized to activate that role and also no role is activated for that object before.
- The activation of a mutually exclusive role for a certain object does matter on the same pattern as it does matter which user has activated which role as “who performed which step on which object” given in [9].
- According to the definition of dynamic separation of duty given in [1], one user can be assigned to all mutually exclusive roles and the user can activate only one of the mutually exclusive roles at the same time. Suppose if the user wants to switch from one mutually exclusive role to another then it can be done only by quitting the previously activated role. This is important to note that if one business task is divided into different subtasks which are executed by different mutually exclusive roles and one user is assigned to all those mutually exclusive roles then one user can activate all roles one by one at different time and execute the whole business task. This is against the concept of separation of duty. So, the user can be authorized to all mutually exclusive roles but it should not be allowed that a user can activate all mutually exclusive roles for the same object.
- The maximum number of activated roles by a user for the same object should always be less than the total number of roles comprised of a complete business process.
- A user should not be allowed to activate two successive dependent roles for the same object. If one role depends on other role and a user is authorized for both of these roles then user should be restricted to activate only one role for the same object.

All the above observations are discussed in greater detail in next Section.

4. Object Based Dynamic Separation of Duty

The newly proposed definition of dynamic separation of duty is given after getting influence from above observations which is contrary to the definition given in [1] as “A user can be assigned to all mutually exclusive roles and is allowed to activate all mutually exclusive roles at the same time but not for the same object and also the user is not

allowed to activate two successive dependent roles for the same object". So, it means a user will be allowed to activate mutually exclusive roles for different objects at the same time. Also a user will not be allowed to activate all mutually exclusive roles for the same object neither at same time nor at different time. After implementing the new proposed definition of dynamic separation of duty, there will be no waste of resources because if a user wants to activate another mutually exclusive role for other object, then that user will be allowed to do that. Also no user will be allowed to execute the whole business task by activating all mutually exclusive roles at different times. So, there will be proper implementation of the concept of separation of duty.

The implementation of newly proposed definition of DSD depends on the proper implementation of role hierarchy, history based separation of duty and operational separation of duty [6, 7, 8, 12]. But we did not discuss these stated concepts in detail.

Nash and Poland proposed a rule in [8] about object based dynamic separation of duty. In this rule they proposed that a user can execute a transaction if the user is authorized to execute that transaction and the user has not executed any other transaction on that object or data item before. We agree to this rule because it supports a part of our newly proposed definition of DSD. There are some points left for discussion like there may be a maximum limit for a user to activate a specific number of mutually exclusive roles. A user can activate a role for an object if the user is authorized to activate that role and user has not reached the maximum limit of activated mutually exclusive roles.

While discussing an example, we will refer the reader to the example given in [8], there are three roles like enter, verify and authorize. Also there are three users like clerk, officer and supervisor authorized for above roles sequentially. All these roles are mutually exclusive to each other. As we discussed earlier that object based DSD is a composite concept which requires the proper implementation of role hierarchy, history based separation of duty and operational separation of duty. The role "enter" is assigned to clerk, the role "verify" is assigned to officer and the role "authorize" is assigned to supervisor. As per operational separation of duty a user is not allowed to activate all mutually exclusive roles for the same object neither at the same time nor at other time. In this way one user will not be able to execute the whole business process itself [12]. According to the role hierarchy "clerk" is the junior most user assigned to role "enter", officer is at the middle level authorized for both roles "enter" and "verify" and at last the supervisor is the senior most user authorized for all roles "enter", "verify" and "authorize".

According to our proposal the user is not allowed to activate two successive dependent roles for the same object. For instance in this example the officer is authorized for the roles "enter" and "verify". The role "verify" depends on the role "enter". So, if the officer activates role "enter" then the other dependent role "verify" should not be allowed to be activated by the officer for the same object. Because it does not make sense that a user verifies its own action. On the same pattern the supervisor is authorized to activate all three roles but the supervisor should not be allowed to activate successive dependent roles for the same object. In this example the supervisor is either allowed to activate the roles "enter" and "authorize" for the same object. If the supervisor activates the role "verify" then the supervisor should not be allowed to activate the roles "enter" and "authorize". So there should be proper record keeping of the roles activated by the user for the object [11] called the history based separation of duty [7].

5. Examples

Different examples have been described where we differentiate the after affects of implementing newly proposed definition of DSD and the definition given in [1]. Also we try to elaborate the newly proposed definition of the dynamic separation of duty as effective, practical and dynamic in nature.

5.1. Case 1

In this case we suppose that there are two mutually exclusive roles R1 and R2. Role R1 and R2 comprise one complete business task. User U1 is already assigned to both roles. In fig.1 the user U1 has activated role R1 for object O1.

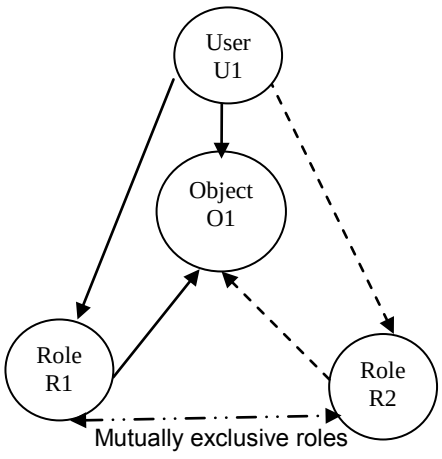

Figure 1. Single user / object with multiple roles

Suppose we follow the definition of dynamic separation of duty given in [1], according to the Figure 1, if the user U1 wants to activate role R2 for the same object O1 then the role R2 can be activated

by quitting role R1 and activating role R2. In this way the User can execute the whole business task which is against the concept of separation of duty.

Table 1. Multiple roles activation to single user with single object

Role	User	Object	Activation
R1	U1	O1	Granted
R2	U1	O1	Denied

If we follow the newly proposed definition of dynamic separation of duty then the user U1 cannot activate role R2 at same time or at some other time shown in Table 1. According to the newly proposed definition of dynamic separation of duty, the concept of separation of duty will be implemented properly. In Table 1, the activation of role R1 to user U1 for object O1 is granted but the activation of role R2 to user U1 for object O1 is denied.

5.2. Case 2

In this case we suppose that there is one user U1 who is assigned to two mutually exclusive roles R1 and R2 comprise one complete business task. There are two objects involved in this case also. The user U1 is already assigned to both roles. The user U1 has activated role R1 for object O1.

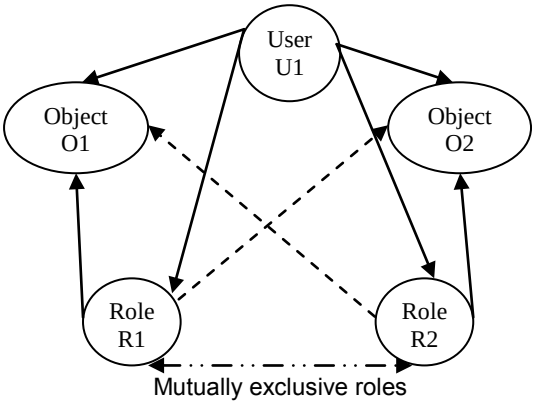

Figure 2. Single user with multiple objects / roles

Suppose we follow the definition of dynamic separation of duty given in [1] then if the user U1 wants to activate role R2 for the object O2 at the same time, then the user U1 will not be allowed to activate role R2 for object O2 because roles R1 and R2 are mutually exclusive to each other that is why they can not be activated by the same user at the same time as shown in Figure 2.

Also if the user U1 wants to activate role R2 for the same object O1, then the user U1 will quit its previously activated role R1 and will activate role R2 for the object O1.

If we follow the newly proposed definition of dynamic separation of duty, if the user U1 wants to activate the role R2 for some other object O2, then the User will be able to activate that role as shown in Table 2.

Table 2. Multiple roles activation to single user with multiple objects

Role	User	Object	Activation
R1	U1	O1	Granted
R2	U1	O2	Granted
R2	U1	O1	Denied
R1	U1	O2	Denied

Also if the user wants to activate role R2 for the same object O1, then it will not be allowed to activate this role for object. So, we believe that this is a practical and dynamic approach.

5.3. Case 3

In this case multiple users, roles and objects are involved. We suppose that there are two users U1 and U2 are assigned to mutually exclusive roles R1 and R2 which comprise one complete business task. There are three objects O1, O2 and O3 involved in this case also. The user U1 has already activated role R1 for object O1 and the user U2 has activated role R2 for object O2.

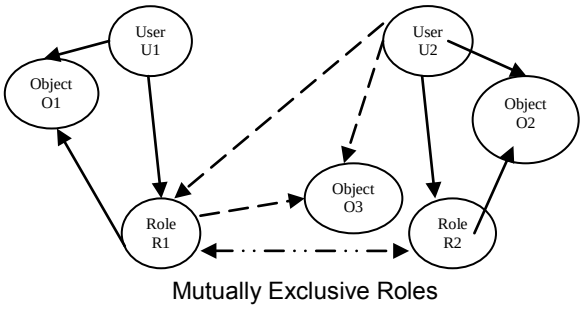

Figure 3. Multiple users with multiple objects / roles

Suppose we follow the definition of dynamic separation of duty given in [1], given that if the user U2 wants to activate role R1 for an other object O3 at the same time then the user U2 will not be allowed to activate role R1 for object O3 because both roles R1 and R2 are mutually exclusive roles and can not be activated by the user U2 at the same time as shown in Figure 3. Also if the user U1 wants to activate role R2 for the same object O1, the user U1 can do this by quitting its previously activated role R1 and activating new role R2. On the same pattern, if the user U2 wants to activate role R1 for same object O2, the user U2 can do this by quitting its previously activated role R2 and activating new role R1.

Table 3. Multiple roles activation to single user with multiple objects

Role	User	Object	Activation
R1	U1	O1	Granted
R2	U2	O2	Granted
R1	U2	O3	Granted
R1	U2	O2	Denied

If we follow the newly proposed definition of dynamic separation of duty given that if the user U2 wants to activate role R1 for object O3, then the user U2 will be allowed to do this as shown in Table 3. If the user U1 wants to activate role R2 for object O1 then the user U1 will not be allowed to do that. Also on the same pattern if the user U2 wants to activate role R1 for the object O2, then the user U2 will not be allowed to do that.

According to the above given cases we have tried to emphasize that we cannot implement the concept of separation of duty by implementing the definition of dynamic separation of duty given in [1]. With newly proposed definition of dynamic separation of duty we can implement the real spirit of the concept of separation of duty.

6. Conclusion

The implementation of RBAC with newly proposed definition of dynamic separation of duty can resolve many problems which we discussed in the observations Section. No single user will be allowed to execute the whole business process. The resources can be maximally utilized and the concept of separation of duty will be implemented in a better way in the form of effectiveness and dynamicity. This will be more dynamic and practical approach as compared to formal definition of dynamic separation of duty given in [1]. We are currently in the phase of implementing the newly proposed definition of dynamic separation of duty.

7. References

[1] American National Standard for Information Technology – Role Based Access Control, ANSI INCITS 359-2004.

[2] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, Charles E. Youman, "Role-Based Access Control Models," Computer, vol. 29, no. 2, pp. 38-47, Feb. 1996, doi:10.1109/2.485845.

[3] Ravi Sandhu and Venkata Bhamidipati, The ASCAA Principles for Next-Generation Role-Based Access Control. Proc. 3rd International Conference on Availability, Reliability and Security (ARES), Barcelona, Spain, March 4-7, 2008, pages xxvii-xxxii. Presentation Keynote Lecture.

[4] Jörg R. Mühlbacher, Christian Praher - DS RBAC - Dynamic Sessions in Role Based Access Control; in: Journal of Universal Computer Science, Vol. 15, Issue 3, pp. 538-554, ISSN 0948-695x (2009).

[5] Gligor, V., Gavrila, S., and Ferraiolo, D. On the formal definition of separation-of-duty policies and their composition. In Proceedings of 1998 IEEE Symposium on Research in Security and Privacy (Oakland, California, 1998), pp. 172–183.

[6] Crampton, J. 2003. Specifying and enforcing constraints in role-based access control. In Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (Como, Italy, June 02 - 03, 2003). SACMAT '03. ACM, New York, NY, 43-50. DOI= http://doi.acm.org/10.1145/775412.775419

[7] Simon, R., and Zurko, M. Separation of duty in role-based environments. In Proceedings of 10th IEEE Computer Security Foundations Workshop (Rockport, Massachusetts, 1997), pp. 183–194.

[8] M.J. Nash and K.R. Poland. Some conundrums concerning separation of duty. In Proceedings of the Symposium on Security and Privacy, pages 201–207, 1990.

[9] Schaad, A., Spadone, P., and Weichsel, H. 2005. A case study of separation of duty properties in the context of the Austrian "eLaw" process. In Proceedings of the 2005 ACM Symposium on Applied Computing (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM, New York, NY, 1328-1332. DOI= http://doi.acm.org/10.1145/1066677.1066976.

[10] Luigi Giuri, Pietro Iglio, "A Formal Model for Role-Based Access Control with Constraints," csfw, pp.136, Ninth IEEE Computer Security Foundations Workshop, 1996.

[11] Sandhu, R. Transaction control expressions for separation of duties. In Proceedings of 4th Aerospace Computer Security Conference (Orlando, Florida, 1988), pp. 282–286.

[12] Ferraiolo, D., Cugini, J., Kuhn, D. R. "Role-Based Access Control (RBAC): Features and Motivations" Proc. 1995 Computer Security Applications Conference, 241-248, December 1995.

[13] Kuhn, D. R. 1997. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In Proceedings of the Second ACM Workshop on Role-Based Access Control (Fairfax, Virginia, United States, November 06 - 07, 1997). RBAC '97. ACM, New York, NY, 23-30. DOI= http://doi.acm.org/10.1145/266741.26674

