A Paradigm for Dynamic and Decentralized Administration of Access Control in Workflow Applications

Andreas Mattas¹, Ioannins Mavridis², and Iason Pagkalos¹

¹Informatics Laboratory, Computers Division, Faculty of Technology, Aristotle University,
54006, Thessaloniki, Greece,

mattman@gen.auth.gr, pangelos@gen.auth.gr

²Department of Applied Informatics, University of Macedonia,
156 Egnatia Street, 54006, Thessaloniki, Greece,
mavridis@uom.gr

Abstract. The administration of authorizations in modern Web-based computing environments has become a primary concern. Application security is characterized by a significant complexity, due to the large number of variations and combinations of objects and operations to be protected. Thus, there is a need for data, processes and context parameters, like time and location, to be combined into a security model that ensures correct decision-making for access. Moreover, access control must often be based on dynamic functional requirements that are capable of embedding the required context information to express application-level access control policies in new application domains, as for example Internet workflow applications. In this work a new paradigm of dynamic and decentralized administration of access control that is based on the DARBAC model is presented. DARBAC concerns access control for a wide-range of collaborative applications and aims to provide fine-grained and dynamic administration of authorizations. The presented implementation assumes Web-based applications to support enforcing of access control at a distributed platform level, and it demonstrates in a step-by-step basis the construction of DARBAC components and their management during run-time.

1 Introduction

Access control ensures in general that accesses to information system's objects occur according to the modes and rules fixed by the corresponding security policies [1]. Subjects request for operations on objects. Their requests are permitted or denied usually by one (or more in distributed systems) reference monitor that is responsible for mediating subject's actions on objects and implementing a particular security policy based on an access control model.

In modern application systems, access control must be based on strictly controlled permissions that are available (through activated roles) just-in-time, only to proper users and until they accomplish specific tasks. To support such control activities, proper mechanisms are needed, which must not overload administrative tasks but

Please use the following format when citing this chapter:

Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

support them in a flexible and effective manner. Access control administration depends mainly on the number of the components that are handled, a parameter that depends on the simplicity of the access control model, as well as to the frequency permissions have are changed. On the other hand, it is the nature of information systems that must reflect the changes of the organization they are serving.

Beyond the traditional mandatory (MAC) [2], [3] and discretionary (DAC) [4], [5], [6] approaches, Role-based access control (RBAC) [1], [7] has received considerable attention and has already been implemented in various operating and database management systems [8] as a convenient way to address the problem of organizing permissions. RBAC's main advantage is the efficient administration of permission assignment to subjects through roles and their hierarchies. Researchers [9], [10] have argued however that one of the main omissions of the RBAC model is the authorization of administration, as well the administration in large scale systems, which must be decentralized and be managed using administrative roles. The ARBAC97 [10] model demonstrates how this can be accomplished using RBAC96 [1] as the underlying model. It is desirable however to develop new standards in the area of access control administration, because it is often the place where security breaks. Other alternate paradigms that have been recently discussed in the literature [10], [11], [12], [13] also address RBAC administration from one point of view.

As Web applications become increasingly available today throughout Internet and corporate intranets, a significant administrative overload is being produced when determining the proper authorization schema [14]. Moreover, additional access control mechanisms are needed to record dynamic changes in the content and context of information that differentiate users' need-to-know requirements, to monitor the system's security state, and to facilitate the carrying out transactional activities [15]. This means that access control models must provide efficient administration of authorizations to ensure that valid users exercise their privileges only during the progress of an official activity of their organization. As a result, new approaches of access control to meet the challenges of these new models of computing are required. An approach in this direction, which is based on the DARBAC (Dynamically Administering Role Based Access Control) model, is presented in this work. More specifically, a paradigm for implementing the DARBAC model in a typical workflow application is demonstrated.

2 Access Control Administration

2.1 Main Approaches - Limitations

The various access control approaches, when examined from an administrative point of view, can be grouped into two main styles of handling subjects: the identity-based and the attribute-based. Traditional identity-based access control models are those also-known as supporting DAC. They restrict access to objects based on the identity of subjects. However, providing fine-grained specification of authorizations on subject's identity entails a significant administrative overload, as the number of

subjects becomes huge. For this reason, decentralized administration was selected, where the user who creates an object (its owner) is the one who decides for the authorizations of other users on accessing it, as well as for any possible delegations of his administrative privileges to specific users (named controllers). Such a rather simple model is not however always sufficient for access control administration in modern distributed and collaborative applications, especially for large and dynamically changing organizations. This is mainly due to the lack of supporting dynamic changes of access rights, associating them to subject's credentials when performing an operation, as well as relating them to attribute of resources or other contextual information [16]. Furthermore, the unlimited freedom of owners to delegate administrative privileges often results in a complicated and potentially uncontrolled administration of authorizations. On the other hand, differentiated protection of objects [17] demands for non-stop access control administration that tends to be costly and prone to error.

The use of groups in common implementations of access control lists (ACLs), was an attempt to make their management easier. However, it still remains the question how groups are formed. A widely accepted answer was to define groups based on the subject's responsibility or job function in the organization, which lead to the notion of roles. Roles (RBAC [1], [7]), as well as labels (MAC [2], [3]), are used in attribute-based access control approaches, where usually a centralized style of administration is used. The use of attributes, like roles in RBAC, provides a significant flexibility as far as it concerns the management of permission assignments to subjects. Roles differ from groups because of the role hierarchies and the inheritance they support. A major drawback of traditional RBAC models, which limits their usefulness, is their inability to take into account fine-grained information from the execution context [18]. Pure role-based access control [1] seems to be suitable for session-based operations that take place in function-oriented organization structures usually used in relatively stable environments. The use of roles for grouping authorizations is based however on general organizational terms without incorporating user, object or process attributes [17]. They also lack of fine-grained administration of authorizations, which are specified based on role and object types, while in collaborative applications detailed control on individual role and object instances is often the case. Even the mechanism of roles in RBAC is characterized as static, as opposed to the dynamic changes of context during the progress of collaborative activities [16]. Furthermore, the concept of session in RBAC restricts its applicability in large distributed and cooperative applications where many users are acting asynchronously but under conditions and constraints that are related to the same workflow or mission.

2.2 Design-time and Run-time Administration

The overall administration problem can be separated into the design-time and run-time access control administration. Design-time administration refers to the transformation of organization policies into static components and mechanisms of the access control system. For example, in Core RBAC [7], design-time access control administration (also known as role engineering [19]) may involve specifying roles and permissions, and their assignments. During run-time, users are defined and assigned to roles in order to execute their tasks. Administration of access control

during runtime introduces more complex issues, which are related to the need for supporting least-privilege, privacy protection and just-in-time access control. For example, a teller can always register a check but he must take first the approval of his manager in order to proceed with payment. So, the teller's permission set has to be initially poor and be enriched only when needed.

In known access control administration models, as for example the ARBAC family of models ([10], [11]), the main effort was focused on design-time to control distributing of administrative authority in a hierarchical way via administrative roles. Only management of user to role assignment (e.g. URA97, [10]) can be considered as happening in run-time. So, there is no concern on run-time administration of permissions and roles; and runtime means plenty of dynamic factors, like the ability to describe protection of objects depending on the processes accessing them [17].

2.3 Passive and Active Security

The above distinction between design-time and runtime administration has also been expressed as the separation between passive and active security approaches [20], [21]. In passive security models primary concern is given to maintaining permission assignments, providing so a static protection of objects. Trying to differentiate the supported protection, access control administration workload becomes too heavy, because many applications utilize static ACLs that are specified in a process that is characterized as labor intensive, costly and error prone.

In active security models there is a clear distinction between assignments and activations of permissions. Moreover, permissions are activated and deactivated depending on the evolution of the tasks or contextual conditions. For this reason, it is desired the contribution of contextual information that is related to subjects (e.g. place and time of access request, identity and roles) or objects (e.g. storage location, attribute values) to differentiate decisions for access. This is done considering not simply user's need-to-know requirements but the need-to-do requirements of the enterprise mission in which the user is participating as member of a team; a process that requires a significant administrative workload during run-time. So, the point is to eliminate this manual procedure by incorporating automatically driven acquisition and processing of live information of context, goal, environment, etc. Such information must be taken under consideration when specifying the least-privilege that a user needs at a specific time to accomplish a particular task in the context of a workflow or mission. However, this may also involve significant administrative workload, especially due to the complexity of mechanisms needed.

Thomas and Sandhu [20] have argued the need for active authorization models, which are self-administrative. Such models should provide automatically the ability to manage the components of an access control system during run-time. Relative research work that has been developed on this direction considers mainly the definition of temporal constraints [22], [23], [24], [25], while TBAC [20] and TMAC [21] models do not provide particular ways for implementing the concepts they have introduced and which have influenced other researchers. Such a development was the introduction of the C-TMAC model [25], where specific procedures for filtering permissions based on the contextual information are proposed. However, it lacks self-administration and fine-grained administration of entities and assignment relations, as

well as multidimensional definition of collaborative contexts [16]. Finally, the PBDM model [26] also addresses the problem of delegation of authority, dealing with the different kinds of delegation: user-to-user, user-to-role, role-to-role. Nevertheless, it introduces a lot of administrative intervention for controlling those delegations.

3 The DARBAC model

A recent effort to form an integrated solution to the above issues has resulted to the introduction of the DARBAC model [27]. The DARBAC model is based on the RBAC [7] and PBDM [26] families of models. It gives priority to the administration of access control during run-time and addresses the problem from a different perspective. More specifically, it considers a distributed Web-based information system wherein a number of missions with possible relationships between them are in progress. The protection state of the system is concerned to be partitioned in particular states that are related to each one mission instance. Moreover, sessions are replaced by the notion of mission instances, which are more suitable for Web-application environments, in order to support enforcement of access control at a distributed platform level. The concept of session, which is used in RBAC-based access control models as a unity wherein the activated roles and permissions, as well as the results of enforcing constraints, are defined on a user basis. A mission instance encompasses the whole effort of a team of users that aims to accomplish a goal through particular tasks and in specific limits (e.g. time, etc). So, various constraints can be enforced regarding specific object-instances, without allowed to users to violate them via multiple sessions. Mission instances are used as a basis for controlling the activation of permissions from subsets of roles to which a user belongs. While sessions are user-oriented, missions are goal-oriented. As a result, the protection state can be checked on collaborative basis rather than on a single user basis. Mission instances can be thought as loose variations of tasks, where instead of operations can be represented states, teams, or co-operations between groups of users. An overview of the DARBAC model is shown in Figure 1.

Fig. 1. The DARBAC model.

The DARBAC model aims to satisfy requirements for access control in workflow systems [28], [24] that include the use of task as a factor for differentiating access

control, distinction between role and process instance-based user group, dynamic authorization based on access history, privacy protection through user exclusion, permission propagation through temporal role delegation, activation of permissions only during the execution of a task and ability to handle temporal constraints.

4 The DARBAC Structure

The DARBAC model relies on six sets of entities called users, organization and mission roles, permissions, missions and objectives (Figure 1). A *user* is a person that uses a Web-application. *Permissions* are modes of access users can exercise on one or more data objects. In addition, DARBAC incorporates the *role* concept in a slightly different manner. More specifically, it distinguishes between organization-roles (job functions within the context of an organization with some associated semantics, regarding authorities and responsibilities conferred on users assigned to them) and mission-roles (temporarily activated or delegated, and serving the specific needs of a given application). Mission-roles are activated only in the frame of mission-instances. A *mission* is considered as a type of project/process that is carried out by a user or a team/group of users with appropriate roles. In collaborative environments, missions can be thought of as identical to teams of users, while in workflow ones a mission may represent a part (tasks) or an entire workflow. The structure of a mission may represent a temporary project team organization (e.g. adhocracy organization [29]); for a finite period of time the members of project team are acting in order to accomplish specific goals that are determined by a number of objectives. When goals are accomplished, the particular mission instance is ended. The notion of *objective* is similar to that of context in [21], but it is used in a broader sense; an objective may include the particular target/object of an operation. For example, the payment of a check concerns a given check that is identified by the check's number. In general, an objective instance may contain values of time intervals, object identities, and other contextual information that contributes to define a restricted range of exercising generally applied permissions. *Permissions* are assigned to mission-roles with a many-to-many Permission-to-Mission-Role Assignment relation (PMRA). Users are also related to organization-roles with a many-to-many User to Organization-role Assignment relation (UORA). Furthermore, mission-roles are assigned to organization ones with a many-to-many Mission-role to Organization-role relation (OMRA). On the other hand, objectives are assigned to missions with a many-to-many Objective to Mission Assignment relation (OMA). The set of skills needed for a mission to accomplish its goal is specified through an Organization-role to Mission Assignment relation (ORMA), where each role refers to an autonomous activity of mission, thus giving an alternative way to represent tasks in workflows.

During run-time, users with sufficient administrative permissions can bind an objective instance to a mission instance through a many-to-one Objective-instance to Mission-instance Binding relation (OMB). In addition, as users participate in mission instances, new entries are added in a many-to-many User to Mission-instance Participation relation (UMP). Two types of constraints can be defined by security designers during build-time: Separation of Duty Constraints (SDC) and Join of Duty Constraints (JDC). SDC and JDC are examined during run-time to decide the users'

participation in mission instances. SDC imposes the rule that no user can participate in a mission-instance with more organization-roles than specified. JDC specifies that a user with a given organization-role can participate in a mission instance, only if a second user with another organization-role is already participating or not. In addition, identity-based inclusions or exclusions of users to participate in a mission instance are specified during run-time with User-Mission Constraints (UMC).

A more formal description of the DARBAC model, which are based on RBAC [7] and PBDM [26] families of models, can be found in [30].

5 The DARBAC Decision-making Process

Decision-making in the presence of a user access request is performed in DARBAC using the permissions gained by mission roles (or possible enrichments from delegations) and the particular values of the objective instances bound to the mission instance, where the user currently participates. Decision-making is accomplished in accordance with the following four-step procedure, which is repeated for every user's access request:

Step 1: Reviewing. In order to perform an operation, the user submits an access request through the Web-application, after he has activated either manually (from a picking list), or automatically (through the application) the prerequisite organization-role to participate in the corresponding mission instance. Then,

1. Given the bound objective instance, the mission instance is sought in OMB.
2. In case there is no such mission instance, the access request is denied. Otherwise the user's participation is sought in UMP. If he already participates in the mission instance, the procedure continues with step 3; otherwise it goes to step 2.

Step 2: Participating. To decide whether the user can participate in the mission instance:

1. UORA is checked to confirm that the required organization-role has been assigned to the user. Otherwise, the access request is denied.
2. ORMA is sought to be determined whether the user is permitted to participate in an instance of that mission.

Step 3: Checking. Verification of user's participation is performed by examining constraints SDC, JDC and UMC, as follows:

1. Based on the values of username, organization-role and mission instance, SDC, JDC and UMC constraints are applied.
2. In case the above constraints are satisfied, the user is allowed to participate in the specific mission instance and UMP is updated.

This step has to be repeated for every new access request, whether or not the user participates already in the mission instance, in order to verify his participation under current dynamically changing conditions.

Step 4: Matching. The final control of whether the user has the right to execute his access request is performed with following actions:

1. Aggregation of permissions the user gains from mission roles, either assigned to his organization role during build-time or delegated during run-time, and
2. Matching the requested operation to the permissions acquired.

6 An Implementation Paradigm

The implementation of DARBAC in a real life situation is demonstrated in this section with an application example from the banking domain. As discussed already, the access control system to be implemented is comprised of two parts: a build-time module for defining and maintaining the components of DARBAC, and a run-time module that enforces all DARBAC security considerations to control user interactions with the banking system. The process of constructing these modules in corresponding phases is briefly described below.

6.1 Build-Time Phase

Access control design during the built-time is carried out in an off-line state. The organization managers in cooperation with the application developers capture the organization's rules and policies to define, name and construct the components of the access control system. Administrative tasks during build-time are quite similar to those specified in ARBAC family of models [10], [11], [12], [13] that address sufficiently distributed administration of role and permission assignments. The demonstrated implementation is focused on a typical banking transaction, known as the "check payment"

Definition of Entities. The security designer defines two missions: MD, as the default mission for all bank's transactions, and MPC for check payments. Additionally, he defines the objective *Work_days*, for the (working) days the banking system is accessible by users, and *Check_ID*, which is the identification number of a check. Then he defines the organization-roles *Teller* (to register and pay checks) and *Manager* (to approve checks' payments), as well as the mission-roles: regular roles *R_teller* and *R_manager*, fixed delegatable role *F_manager* and temporal delegatable role *T_teller* (for role delegation purposes). He also defines *DPCm*, a delegation role owned by *F_manager* (for use in mission MPC). Permissions are distinguished between regular: *Insert*, *Approve*, *Pay*, and administrative ones: *DelegateDPCm*, *StartMPC*, *EndMPC*.

Definition of Relationships. The security designer assigns (OMRA) mission-roles to *Manager* and *Teller*, and permissions (PMRA) to roles, as shown in figure 2. Then he assigns (OMA) objectives *Work_days* and *Check_ID* to missions MD and MPC, respectively. He also assigns (ORMA) roles *Manager* and *Teller* to missions MD and MPC.

Definition of Constraints. SDC and JDC constraints are declared, as follows:

- $SDC_{MPC} = (\{\text{Manager, Teller}\}, 2)$, which means that the same user may activate, only one of organization-roles Manager and Teller when participating in any mission instance of mission MPC.
- $JDC_{MPC} = (\text{Teller, Manager, } ||)$, which means a user with organization-role Manager can participate in any mission instance of mission MPC, only while another user with organization-role Teller is participating.

Fig. 2. OMRA (a) and PMRA (b).

6.2 Run-Time Phase

Access control managing during run-time must be able to cope with situations related to any daily or emergency activities in an organization. Administrative operations are performed by the security administrator, as well as a wide variety of users under certain constraints; e.g. a user can start a new mission instance and bind it to an objective instance. The security administrator assigns organization-roles to users, which can then participate in a default mission instance. In order to perform their specific tasks they have to participate firstly in corresponding mission instances.

During build-time, fewer permissions than necessary are assigned to mission roles. However, temporal role-to-role delegation can be used in run-time to provide further functional dependencies in the context of a mission instance. This is useful to ensure that no job can be completed unless permitted by a supervisor role who delegates the necessary additional permissions (through an appropriate delegation role [26]). The type of delegation adopted in DARBAC is temporal, since its term ends automatically when a user with administrative permissions terminates the mission instance. This fact, in conjunction with the required participation of users in mission instances for being able to activate mission roles, results in the revocation of available permissions, either enriched via delegations or initially assigned ones.

A demonstration of the progress of this example during the run-time is depicted in figure 3 where it is assumed that the time is advancing from left to right. Mission-instances are depicted with different shades. Particular operations accomplished during this example are also distinguished with vertical lines.

Fig. 3. Progress of access requests through mission instances.

System initialization. First of all, the security administrator starts up the instance MD#15 of the default mission MD, and binds it (OMB) to the instance (Monday..Friday) of objective Work_Days. Then, he creates user accounts Alice and Bob and assigns them (UORA) to roles Manager and Teller, respectively. Both users can use the banking system (after participate in MD#15) only during working days.

Check registration. Let's assume that Bob has received a check with identification number '960' and proceeds with the check's registration. Check's registration includes two operations: an administrative one (StartMPC) and a regular one (Insert), as follows:

- StartMPC: Bob participates in MD#15 with organization-role Teller and executes operation 'StartMPC' that starts a new mission instance of MPC (e.g. MPC#45) and binds it to an instance of objective Check_ID with value (960).
- Insert: Bob participates in MPC#45 (constraints SDC_{MPC} and JDC_{MPC} are satisfied) as Teller and executes operation 'Insert' to register the check.

Payment approval. Then Bob asks his manager (Alice) to approve the check's payment. Check's approval includes a regular operation (Approve) and an administrative one (DelegateDPCm), as follows:

- Approve: Alice participates in MD#15 and MPC#45 with organization-role Manager (JDC_{MPC} is satisfied because Teller already participates) and executes operation 'Approve'. However, user Bob cannot complete the payment until the permission Pay will be given to him with a role-to-role delegation.
- Delegate_DPC: Alice (still participating in MPC#45) assigns mission-role DPCm to mission-role T_teller (due to administrative permission DelegateDPCm).

Check payment. Bob can now proceed with the check's payment, as follows:

- Pay: Bob (still participating in MPC#45) executes operation 'Pay' that results in completing the check payment.
- EndMPC: Bob terminates MPC#45, giving also an end to any temporal administrative operation that took place during that mission instance. As a result Teller's extended rights are over and his first assigned permissions are restored.

As a result of the above process, it has been possible to apply fine-grained and just-in-time access control using only administrative functions of the access control system. Hence, access control can be clearly distinguished from application logic. Moreover, users can exercise administrative permissions in a really distributed environment.

7 Conclusion

In this paper the application of dynamic administration of role-based access control based on the DARBAC model is demonstrated. DARBAC guides the security designer during build-time and provides an integrated framework for decentralized and temporal administrative activities during runtime.

The DARBAC model is suitable for access control in a wide-range of collaborative applications, usually set-up in Web-based environments. It introduces a distinct structure for defining its components, attributing administrative privileges to a variety of roles and imposing constraints for separation, synchronization and user-based inclusion/exclusion to satisfy the security principles of least-privilege, separation of duties, conflict of interests and privacy protection. DARBAC also preserves the advantages of permission administration that RBAC models offer and yet introduces the concept of mission, in addition to that of roles, as an abstract mechanism to formulate the objective information, which in turn identifies the aim and context of activities to be performed by a group/team of users with prerequisite roles. Dynamic administration is achieved by granting to normal and administrative users the proper administrative permissions to manage missions and their objectives, to perform temporary delegations and to apply constraints governing mission participations; allowing so controlled transfer of role competences and decentralization of authority.

A paradigm of dynamic and decentralized administration of access control based on the DARBAC model has been presented, through an implementation of DARBAC in a real life situation application example from the banking domain. The presented implementation assumes Web-based applications to support enforcing of access control at a distributed platform level and it demonstrates in a step-by-step basis the construction of DARBAC components and their management during run-time. It can therefore be said that DARBAC can provide a promising alternative and an interesting starting point for further research work on new access control administration paradigms for Web applications and services. Future work includes further study on relationships between Web applications and missions, as well as exploiting additional features of Web-services for access control across different administration domains.

References

1. Sandhu, R.: Role-Based Access Control. *Advances in Computers*. Academic Press (1998).
2. Bell, D.E., LaPadula, L.J.: Secure computer systems: mathematical foundations and model. Technical Report, MITRE (1974).
3. Biba, K.J.: Integrity Considerations for Secure Computers Systems. Bedford, MA: The MITRE Corporation (1977).
4. Lampson, B.W.: Protection. In 5th Princeton Symposium on Information Science and Systems, p. 437-443. Reprinted in *ACM Operating Systems, Review* 8(1) (1971) 18-24.
5. Graham, G.S., Denning, P.J.: Protection - principles and practice. In *AFIPS Spring Joint Computer Conference* (1972) 40:417-429.
6. Harrison, M., Ruzzo, W., Ullman, J.: Protection in operating systems. *Communications of the ACM* 19, 8 (1976) 461-471

7. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: A Proposed Standard for Role Based Access Control. *Transactions on Information and System Security* (2001)
8. Chandramouli, R., Sandhu, R.: Role Based Access Control Features in Commercial Database Management Systems. 21st National Information Systems Security Conference (1998)
9. Sandhu, R.: Future Directions in Role-Based Access Control Models. In conference: International Workshop MMM-ACNS. St. Petersburg, Russia (2001)
10. Sandhu, R., Bhamidipati, V., Munawar, Q.: The ARBAC97 Model for Role-Based Administration of Roles. *Transactions on Information and System Security* (1999) 105-135
11. Sandhu, R., Munawar, Q.: The ARBAC99 Model for Administration of Roles. *Proceedings of the 15th Annual Computer Security Applications Conference* (1999) 229
12. Oh, S., Sandhu, R.: A Model for Role Administration Using Organization Structure. 7th ACM Symposium on Access Control Models and Technologies. USA (2002) 155-168
13. Kern, A., Schaad, A., Moffett, J.: An Administration Concept for the Enterprise Role Based Access Control Model. 8th Symposium on Access Control Models and Technologies (2003)
14. Kooker, R., Kane, S.: Identity Management: Role Based Access Control for Enterprise Services. *Command and Control Research and Technology Symposium*. USA (2004)
15. Joshi, J., Aref, W.G., Ghafoor, A., Spafford, E.H.: Security Models for Web-Based Applications. *Communications of the ACM*, Vol. 44. No. 2. (2001)
16. Tolone, W., Ahn, G., Pai, T., Hong, S.: Access control in collaborative systems. *Source ACM Computing Surveys (CSUR) archive*, Vol. 37. No. 1. (2005) 29 – 41
17. Kern, A., Kuhlmann, M., Kuropka, R., Ruthert, A.: A Meta Model for Authorizations in Application Security Systems and their Integration into RBAC Administration. 9th ACM Symposium on Access Control Models and Technologies. USA (2004)
18. Yao, W., Moody, K., Bacon, J.: A Model of OASIS RoleBased Access Control and its Support for Active Security. In *SACMAT'01*, Chantilly, Virginia, USA (2001)
19. Epstein, P., Sandhu R.: *Engineering of Role/Permission Assignments*. 17th Annual Computer Security Applications Conference (2001)
20. Thomas, R., Sandhu, R.: Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization Management. In *Database Security, XI: Status and Prospects* (eds. T.Y.Lin and S. Qian), Chapman and Hall, London (1997)
21. Thomas, R.: Team-Based Access Control: A Primitive for Applying Role-Based Access Controls in Collaborative Environments. 2nd ACM Workshop on RBAC, USA (1997)
22. Bertino, E., Bonatti, P.A., Ferrari E.: TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security (TISSEC)*. (2001) 191 – 233
23. Joshi, J., Bertino, E., Latif, U., Ghafoor, A.: Generalized Temporal Role-Based Access Control Model. *IEEE Transaction on Knowledge and Data Engineering*. (2005) 4 - 23
24. Atluri, V., Huang, W.: An Authorization Model for Workflows. In *Lecture Notes in Computer Science*, No.1146. Springer-Verlag (1996) 44–64
25. Georgiadis, C., Mavridis, I., Pangalos, G., Thomas, R.: Flexible team-based access control using contexts. 6th ACM Symposium on Access Control Models and Technologies. (2001)
26. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. *Proceedings of 8th ACM Symposium on Access Control Models and Technologies*, Como, Italy (2003)
27. Mattas, A., Mavridis, I., Pangalos, G.: Towards Dynamically Administered Role-Based Access Control. 14th Int. Workshop on Database and Expert Systems Applications. (2003)
28. Wu, S., Sheth, A.P., Miller, J.A., Luo, Z.: Authorization and Access Control of Application Data in Workflow Systems. *Journal of Intelligent Information Systems (JIIS)*. (2002) 71-94
29. Shim, W.B., Park, S.: Toward an Improved RBAC Model for the Organic Organization. 9th International Conference on Parallel and Distributed Systems, Taiwan (2002)
30. Mattas, A., Mavridis, I., Pangalos, G.: The DARBAC Model, Technical Report INFOLAB-TR01-2005 (2005) URL: <http://infolab.gen.auth.gr/TR/INFOLAB-TR01-2005.pdf>