Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

An Access Control Model for Workflows Offering Dynamic Features and Interoperability Ability

Shih-Chien Chou and Chien-Jung Wu
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 974, Taiwan
E-Mail: scchou@mail.ndhu.edu.tw

Abstract - *Workflow management systems (WFMS) are useful in designing and evolving processes such as business processes. Recently, workflow security has been recognized as important. Workflow security issues include network security, authentication, access control, and so on. Our research focuses on access control. This paper proposes a model WfRBAC (role-based access control within workflows) for workflow access control. WfRBAC is an extension of the important RBAC milestone RBAC96. Our research revealed that the dynamic features of managing dynamic role switching, managing dynamic role association change, and preventing indirect information leakage are essential in workflow access control. In addition, interoperability ability to exchange access control information among workflows, and handling constraints are also necessary. Since RBAC96 fails to offer the dynamic features and interoperability, WfRBAC adds mechanisms for them. This paper presents WfRBAC.*

Keywords: Workflow, access control, security, role-based access control (RBAC)

1. Introduction

Workflow management systems (WFMS) are useful in designing and evolving processes such as business processes. Generally, a business process is composed of workflows; a workflow is composed of tasks; and a task is composed of activities [1]. During workflow execution, an activity is generally activated by events (e.g., the event “a customer order an item” activates an activity to handle the order). When an activity is activated, the activity is enacted by a *role* (e.g., a manager and an operator). A role may be played by a human being or an automatic system. When enacting an activity, a role may consume *data* (e.g., customer order) and may produce data and events to activate other activities. Moreover, a role may use *resources* (e.g., order management system) when enacting an activity.

Recently, workflow security has been recognized as important. Some (but not all) important workflow security issues are listed below:

a. Network security. Quite a few business processes operated on the network, which results in information exchange on the network. Information on the network should be protected. Generally, network security can be achieved through cryptography [2].
b. Authentication. Authentication gains confidence that the person assigned a task is the right person that should be assigned the task. This security can be achieved through passwords, digital signatures, and so on.
c. Access control. Access control refers to granting/revoking access rights. It controls the access of resources, information, and so on. Access control mechanisms prevent information leakage within a workflow.

We focus on the research of access control. Approaches used in the control include access control matrix [3-5], Chinese wall [6], role-based access control (RBAC) [7-9], and so on. We apply RBAC in our research. Below we give a brief introduction to RBAC.

In RBAC, a role is a collection of permissions [10]. A role can establish one or more sessions. During a session, a user playing a role possesses the permissions of the role. Role assignment is based on user responsibilities. That is, the role assigned to a user should possess permissions to facilitate finishing the user’s responsibility. When a user finishes his responsibility or a session ends, the role assignment will be removed, which results in revoking permissions from the user. Users can change role during his responsibility if necessary. This facilitates enforcing the need-to-know principle [11-13]. A major advantage of RBAC is that permissions are bound to roles instead of users. With this, adjustment of user permissions can be achieved through role assignment.

An important milestone of RBAC is RBAC96 [14], which was widely applied in access control [7-9, 15-18]. In RBAC96, the assignment between roles and permissions and that between users and roles can be changed during a session. Nevertheless, the change can only be accomplished through the intervention of human beings, who are generally the system administrators. That is, the change is static [19]. If RBAC96 is applied to workflow access control, both user-role assignment and role-permission assignment cannot be dynamically changed during the execution of a workflow. Since a workflow’s access control policy may be dynamically changed [4, 19-20], RBAC96 is insufficient in workflow access control. We identified that RBAC96 fails to handle the following important dynamic features:

a. Dynamic role switching. The switching refers to changing user-role assignment according to events occurred during workflow execution. For example, suppose a company will automatically promote a real world customer to a VIP when the customer’s total consumption amount passes a threshold (here we suppose the promotion is accomplished by an automatic checking system without the intervention of human beings). In this case, the role played by the real world customer will be dynamically switched from "customer" to "VIP" during the execution of the workflow. Since role switching may happen anytime during the execution of a workflow and the switching should be handled dynamically, RBAC96 cannot handle the switching because RBAC96 can only change user-role assignment statically (i.e., the change needs the intervention of human beings).

b. Role association management. Associations may exist among real world users. When the users play roles in a workflow, the associations exist among the roles played by the users. Role associations may result in different role permissions. Since role associations may dynamically change according to user association change during the execution of a workflow, role-permission assignment in a workflow may be dynamically changed. We use an example to explain this. Suppose a customer can get a supplementary item when he buys an item from a company, in which the supplementary item is selected from a list. If a customer and a manager of the company are not friends, the customer selects the supplementary item from a default list. On the other hand, if they are friends, an extra list is provided to the customer for the selection. In this example, the permission on the extra list for the customer role is decided by the association "friends" between the roles "manager" and "customer", in which the roles are played by the real world customer and manager. Since a real world customer and a real world manager may become friends or they may break friendship anytime during the execution of a workflow, the permission on the extra list for the customer role may change dynamically.

c. Preventing indirect information leakage. Information may leak through the third one. This situation is called indirect information leakage. We use an example to explain the importance of preventing indirect information leakage. Suppose in a company, an operator is not allowed to know the exact price of an item. When a customer order is received, an operator is required to prepare an order form filled with every thing except the price. The manager then fills the price into the form. The operator can read the form before the filling but cannot after the filling because the operator cannot read the price. In this regard, the access right on the form before the filling is different from that after the filling. To prevent the operator from knowing the price by reading the form, the access rights of the form should be changed dynamically after the manager fills the price.

The three dynamic features mentioned above can be achieved through human intervention (i.e., be achieved statically) in RBAC96 by suspending a workflow and then resuming the workflow. Nevertheless, hundreds of workflow instances may be executing in a company at a time. Handling the dynamic operations through human intervention may be cumbersome and error-prone. Therefore, we propose that dynamically offering of the features by WFMS is necessary.

In addition to the three dynamic features above, exchanging access control information among workflows is also important. Generally, workflows communicate with one another through interoperability. Although many workflow interoperability protocols were proposed [21-24], none of them exchange access control information. In our opinion, information passed from a workflow to another one should be protected just like the information is within the original workflow. Still another feature that should be offered is constraining the components of an access control model. For example, separation-of-duty (SoD) constraints are necessary in access control.

We designed a model WfRBAC (role-based access control within workflows) for access control within workflows based on RBAC96 [14]. Since RBAC96 fails to offer the dynamic features and does not allow exchanging access control information, we developed WfRBAC to offers all the features we need. This paper proposes WfRBAC.

2. Related Work

Traditional access control is achieved by access control matrix (ACM) [3]. According to ACM, a subject can access an object if the required access right is recorded in the matrix. ACM allows only static access control [4, 20]. That is, as long as an access right is recorded in an ACM, the access right can be granted to the subject anytime when the subject needs the right. Since access rights may be changed during the execution of a workflow [4, 19-20], static access control is insufficient. To achieve dynamic access control, many models were proposed. We survey some below.

The DACM (dynamic access control matrix) approach [4] dynamically grants access rights to subjects during the execution of a workflow. Although the granting of access rights is based on an ACM, a subject may be granted different access rights under different situations. This allows dynamic allocation of access rights. Since DACM does not take roles into consideration, it cannot handle dynamic role switching and role associations.

The model in [5] is similar to DACM. It adds the *transition* dimension to the two-dimensioned ACM (here a transition causes an activity to enact and the two dimensions are respectively *subject* and *object*). The model also incorporates the concept of roles. In the model, a role in different transition will be granted different access rights. This makes the access control more precise than static ACM. The drawback of the model is that the three-dimensioned ACM is large and handling the matrix results in substantial overhead. Moreover, the model cannot handle dynamic role switching and role associations.

TBAC (task-based authorization control) [20] associates authorization steps with tasks (or activities). To enact a task, the associated authorization step is first executed to assign the task to a role. Within an authorization step, the allowed roles for a task are placed into a trustee set, from which a role is selected each time the authorization step is executed. An authorization step also consists of protection state, which primarily consists of permissions. Trustee set and protection state can be dynamically changed during the execution of a workflow. Therefore, TBAC allows granting access rights dynamically, which overcomes the drawback of the static ACM or RBAC. Moreover, allowing the trustee set to change dynamically implies that dynamic role switching can be achieved. Nevertheless, since the focus of TBAC is task, it does not take role associations into consideration.

The model in [19] models workflows using Petri-net. Its access control mechanism is based on TBAC. The model improves the feature of static role-permission assignment in RBAC using dynamic assignment. Similar to TBAC, the model achieves dynamic role-permission assignment through authorization steps. Like TBAC, the model can achieve dynamic role switching but cannot manage role associations.

The model in [25] applies RBAC for access control. The focus of the model is handling role constraints, especially dynamic separation-of-duty. Below we use an example to explain dynamic separation-of-duty. Suppose the activity "Evaluate student scores" and "Double check student scores" are all assigned to the role "teacher". Also suppose that the activities cannot be assigned to the same teacher. That is, the person playing the role "teacher" to enact the former activity should be different from the one enacting the latter activity. This constraint is referred to dynamic separation-of-duty. The paper in [25] provides a language to handle dynamic separation-of-duty and other constraints. Nevertheless, in our opinion, dynamic separation-of-duty can be solved by adding roles. For example, the above problem can be solved by splitting the role "teacher" into "evaluationTeacher" and "doubleCheckingTeacher", and then require that the two roles cannot be played by the same human being (this is a kind of static separation-of-duty). As to the feature of handling dynamic role switching and role associations, the model in [25] does not offered.

Napoleon [8] enforces business security policy using multiple-layered RBAC. For example, if an activity needs a permission to enact, Napoleon enforces roles with that permission to enact the activity. Napoleon is a mechanism to enforce security policy instead of a mechanism to define security policy. Therefore, the objective of Napoleon is different from that of our research. In this regard, we cannot say that Napoleon can handle dynamic role switching and role associations.

The model in [9] is the previous version of Napoleon. The difference between the model and Napoleon is that the model in [9] uses only two layers, namely the application designer layer and the system administrator layer [8], whereas Napoleon may use more than two layers.

The MLS (multilevel security) model in [26-27] uses multilevel security to achieve access control. The basic concept of the model is assigning security levels to subjects and objects. A subject cannot access objects in higher security levels. According to security levels, tasks can be split into multiple domains. Tasks in a domain are within the same security level. Activities within a domain can be interacted. Nevertheless, if activities in different domains should interoperate, a security policy should be followed. In the model, activities are assigned to human beings instead of roles. Therefore, user-permission assignment and revocation cannot be as flexible as those in RBAC. Moreover, since roles are not handled in the model, dynamic role switching and role associations cannot be managed.

The model in [28] regulates the access control of inter-organization workflows. It is similar to the MLS model [26-27], which models the workflows of different organizations in different security domains. A workflow in a security domain is executed following the access control policy within the domain. If activities in different domains should interoperate, an inter-organization security policy should be followed. The concept of role is incorporated in this model, which makes the model more flexible than MLS. Nevertheless, dynamic role switching and role associations cannot be handled in the model.

The DW Chinese wall model [6], like the model in [28], controls the access of inter-company information. A subject in a company can access all information within the company. A subject can also access all information that is not considered conflict-of-interest. We use an example to explain conflict-of-interest information below. Suppose a customer wishes to buy an item with a price less than US$ 400. Then, the information "preferredPrice < 400" is considered conflict-of-interest. The rationale is that a company knowing the information can sell the item with a price of US$ 399.9, which results in a frustration for the competing companies. In the model, information access within a company is not controlled because the control granularity is only detailed to the company level. That is, a role in a company can access all information in the company. This implies that dynamic role switching and role associations are not handled in the DW Chinese wall model.

The model in [7] uses RBAC96 as the basic for access control within workflows. Control policies in the model are similar to those in our model. Nevertheless, the model does not handle dynamic role switching and role associations.

WfACL [29] is our previous work. It controls workflow information access using ACLs. Although the model offers all the dynamic features, it cannot exchange access control information among workflows (i.e., no interoperability among workflows is allowed) and cannot handle constraints. Since interoperability is so important that WfMC [30] defined interoperability standards such as Wf-XML [22], WfRBAC can be regarded as a substantial extension of WfACL (note that the interoperability standards offered by WfMC do not include access control information exchange).

3. WfRBAC Model

No standard for workflow definition is available by now (even WfMC [30] does not offer a standard). Therefore, we give definitions to a workflow from our perspective.

A workflow is composed of tasks linked by *dependencies* [6], in which dependencies are also known as *transitions* [4]. Task can be further decomposed into activities linked by dependencies. A transition is associated with data, events, and conditions. The information associated with a transition determines whether an activity can be enacted. In our definition, we do not differentiate tasks from activities. We collectively call them *activities*. Moreover, we do not differentiate events from conditions. We collectively call them *events*.

During workflow execution, activities are enacted by roles. A role may be played by a human being or an automatic system. When enacting an activity, a role may use resources, consumes data, and produces data and events to activate other activities. To prevent workflow information leakage during workflow execution, we embed WfRBAC in a workflow. Here *workflow information* is composed of data and events in a workflow.

WfRBAC controls the access of workflow information during workflow execution. The access of activities is not controlled because ensuring secure activity assignment (i.e., ensuring that an activity is assigned to the right person) is an issue of authentication but not access control. The components of WfRBAC are described below (see [14] to compare the components of RBAC96 with those of WfRBAC):

a. A set of permissions (*P*). A permission is an access right from a role to a data or event.
b. A set of roles (*R*). A role is played by a user. Each role is associated with a set of permissions. The permission set constitutes a capability list [31] for the role.
c. A many-to-many permission to role assignment (*PA*).
d. A set of role associations (*ASO*). *ASO* affects role permissions.
e. A set of users (*U*), which are human beings or automatic systems. Users play roles. A user playing a role possesses the permissions of the role.
f. A set of sessions (*S*). A session corresponds to a workflow instance.
g. A many-to-many user to role assignment (*UA*). A user may play many roles within a session and multiple users may play the same role.
h. A function that maps a session to a single user (*SU*).
i. A function that maps a session to a set of roles (*SR*).
j. A partially ordered role hierarchy (*RH*). If a relationship “x $\geq$ y” exists between the roles “x” and “y”, “x” possesses all the permissions of “y”.
k. A collection of constraints (*CNS*). Currently, WfRBAC models only the *simple dynamic separation of duty* constraints mentioned in [32]. The constraints are modeled using logic expressions. For example, a user in a workflow cannot simultaneously play the roles “customer” and “VIP”. In this case, an exclusive-or relationship exists between the two roles, which forms a logic expression.
l. A set of dynamic actions (*DACT*). The actions handle dynamic role switching, manage dynamic role association change, prevent indirect information leakage, and exchange access control information among workflows.

WfRBAC should be embedded in a workflow to control workflow information access during workflow execution. With WfRBAC embedded, a workflow is composed of two major components, which is: (a) the original workflow (*Wfl*) and (b) the associated access control policy (*Acp*), which are respectively defined below:

Definition 1: *Wfl* = (*EVN*, *DAT*, *RSO*, *RLE*, *ASO*, *ACTV*), in which

a. *EVN* is a set of events, which may be obtained from outside of the workflow or produced within the workflow.
b. *DAT* is a set of data, which may be initially available or produced by the workflow.
c. *RSO* is a set of resources.
d. *RLE* is a set of roles played by human beings or automatic systems.
e. *ASO* is a set of role associations.
f. *ACTV* is a set of activities. An activity *act* in *ScWf* is defined as follows:

act = (*OPR*, *RRL*, *IEV*, *IDT*, *IASO*, *RRS*, *OEV*, *ODT*), in which

OPR is a set of operations that should be performed to finish *act*,
RRL is a set of roles that are assigned the activity *act*; $RRL \subseteq RLE$,
IEV is a set of input events that should occur (i.e., should be true) for the activity *act* to enact; $IEV \subseteq EVN$,
IDT is a set of input data that should be available for the activity *act* to enact; $IDT \subseteq DAT$,
IASO is a set of association information; $IASO \subseteq ASO$. *IASO* is used as input of an activity, which guides the WFMS to identify the permission of a role (we will describe this later).
RRS is a set of resources that are needed to finish the activity *act*, $RRS \subseteq RSO$,
OEV is a set of output events produced by the activity *act*; $OEV \subseteq EVN$, and
ODT is a set of output data produced by the activity *act*; $ODT \subseteq DAT$.

Definition 2. *Acp* = (*InstantiationEvent*, *RH*, *RPER*, *APER*, *DACT*, *CNS*, *XCH*), in which

a. *InstantiationEvent* is a set of events whose values should be assigned when a workflow instance is instantiated. The events generally set up initial role associations and start the instantiated workflow instance.
b. *RPER* is a set of regular permissions for roles.
c. *APER* is a set of association permissions. *RPER* is used when a role is not within a role association whereas *APER* is used when a role is within a role association. For example, suppose a customer and a manager are not friends, the permission of customer should be identified from *RPER*. On the other hand, if they are friends, the permission of customer should be identified from *APER*.
d. *DACT* is a set of actions that offer the dynamic features mentioned in section 1. All the actions are initiated by events. According to page limit, we cannot describe the details of *DACT*.
e. *CNS* is a set of simple dynamic separation of duty (SoD) constraints.
f. *XCH* is a set of information exchange among workflows. Information exchange is achieved through sending and receiving protocol. The sending workflow uses the statement “*sendMessage*(*Wf*, *ARG*)” to send access control information and

arguments, in which *Wf* is the receiving workflow and *ARG* is a list of arguments. To prevent information leakage, every argument passed to other workflows is explicitly associated with its access rights. The access rights of an argument in WfRBAC constitute an access control list (ACL). An ACL is composed of an RACL (read access control list), a WACL (write access control list), and a role association for the ACL to be valid. For example, the ACL of an argument "{manager, president; president; managementGroup}" means that the argument can be read by the roles "manager" and "president", can be written by the role "president", and the read and write access rights are valid in the role association "managementGroup". Note that the ACL of an argument is composed of roles and role associations in the *receiving workflow*, because the arguments are used in the receiving workflow. Although a workflow does not know the details of other workflows, the *workflow programmers* know them. Therefore, defining ACL for arguments in the above manner can be achieved (because workflows are programmed by workflow programmers).

In the information exchange protocol, the receiving workflow uses the statement "*receiveMessage*(*Wf*, *PAR*)" to receive the access control information and arguments, in which *wf* is the receiving workflow and *PAR* is a list of parameters to receive arguments. When an exchange occurs, the parameter receiving an argument receives both the argument's value and ACL so that the parameter can inherit the security level of the argument to prevent information leakage. After receiving the ACLs, the ACLs are transformed into permissions to control workflow information access in the receiving workflow. For example, the ACL example in the previous paragraph will be transformed into the following permissions (suppose the ACL is received by the parameter "par1"): "(par1, manager, R, managementGroup)", "(par1, president, R, managementGroup)", and "(par1, president, W, managementGroup)".

Although the definition of *Acp* does not show all the components of WfRBAC, the missing components do exist. They are managed by the supporting environment. Definition 2 only shows the components that should be explicitly described when embedding WfRBAC in workflows.

4. Conclusions

This paper proposes a model WfRBAC (role-based access control within workflows) for workflow access control, which is based on RBAC (role-based access control). WfRBAC is an extension of the important RBAC milestone RBAC96. In RBAC96, the assignment between roles and permissions and that between users and roles are static. Although both the assignments can be changed during a session, the change can only be accomplished through the intervention of human beings. With this, RBAC96 does not allow the user-role assignment or role-permission assignment to be changed dynamically during the execution of a workflow. Since a workflow's access control policy may be changed dynamically, RBAC96 is insufficient in workflow access control. In fact, we identified that RBAC96 fails to offers the dynamic features of managing dynamic role switching, role association change, and indirect information leakage prevention. Moreover, RBAC96 cannot exchange access control information among workflows. According to the insufficiency, WfRBAC extended RBAC96 by adding event-triggered dynamic actions (DACT) to offers the three dynamic features and the protocol of exchanging access control information.

References

[1] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", *Distributed and Parallel Databases*, vol. 3, pp. 119-153, 1995.

[2] W. Ford, and M. S. Baum, *Secure Electronic Commerce, second edition*, Prantice-Hall, 2001.

[3] M. H. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in Operating Systems", *Communications of the ACM*, vol. 19, no. 8, pp. 461-471, 1976.

[4] M. S. Olivier, R. P van de Riet, and E. Gudes, "Specifying Application-level Security in Workflow Systems", in *Proceeding of the 9'th International Workshop on Database and Expert Systems Applications*, pp 346-351, 1998.

[5] K. Knorr, "Dynamic Access Control through Petri Net Workflows", in *Proceedings of the 16'th Annual Conference on Computer Security Application*, 2000.

[6] V. Atluri, S. A. Chun, and P. Mazzoleni, "A Chinese Wall Security Model for Decentralized Workflow Systems", in *Proceedings of the 8'th ACM Conference on Computing and Communication Security*, 2001.

[7] G. J. Ahn, R. Sandhu, M. Kang, and J. Park, "Injecting RBAC to Secure a Web-Based Workflow System", in *Proceedings of the 5'th ACM Workshop on Role-Based Access Control*, 2000.

[8] C. Payne, D. Thomsen, J. Bogle, and R. O'Brien, "Napoleon: A Recipe for Workflow", in *Proceedings of the 15'th Computer Security Applications Conference*, pp. 134-142, 1999.

[9] D. Thomsen, D. O'Brien, and J. Bogle, "Role Based Access Control Framework for Network Enterprises", in *Proceedings of the 14'th Annual Computer Security Applications Conference*, 1998.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, vol. 29, no. 2, pp. 38-47, 1996.

[11] M. Nyanchama, S. Osborn, "Access rights in Role-Based Security Systems", *Database Security VIII: Status and Prospects*, pp. 37-56, 1994.

[12] R. Sandhu, V. Bhamidipati, Q. Munawer, "The ARBAC97 Model for Role-Based Administration of Roles", *ACM Transaction on*

Information and System Security, vol. 2, no. 1, pp. 105-135, 1999.

[13] D. J. Thomsen, “Role-Based Application Design and Enforcement”, *Database Security IV: Status and Prospects*, pp. 151-168, 1991.

[14] R. Sandhu, “Role Hierarchies and Constraints for Lattice-Based Access Controls”, in *Proceedings of the Fourth European Symposium on Research in Computer Security*, pp. 65-79, 1996.

[15] S. -C. Chou, “Embedding Role-Based Access Control Model in Object-Oriented Systems to Protect Privacy”, to appear in *Journal of Systems and Software*.

[16] S. -C. Chou, “L^{n}RBAC: A Multiple-Leveled Role-Based Access Control Model for Protecting Privacy in Object-Oriented Systems”, to appear in the *Journal of Object Technology*.

[17] K. Izaki, K. Tanaka, M. Takizawa, “Information Flow Control in Role-Based Model for Distributed Objects”, in *Proceedings of the 8’th International Conference on Parallel and Distributed Systems*, pp. 363-370, 2001.

[18] Z. Tari, S. -W. Chan, “A Role-Based Access Control for Intranet Security”, *IEEE Internet Computing*, vol. 1, no. 5, pp. 24-34, 1997.

[19] X. Dong, G. Chen, J. Yin, and J. Dong, “Petri-net-based Context-related Access Control in Workflow Environment”, in *Proceedings of the 7’th International Conference on Computer Suported Cooperative Work in Design*, 2002.

[20] R. K. Thomas and R. S. Sandhu, “Task-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management”, in *Proceedings of the IFIP WG11.3 Workshop on Database Security*, 1997.

[21] WfMC Workflow Standard Interoperability -- Internet e-mail MIME Binding, Document Number WFMC-TC-1018, http://www.wfmc.org/standards/docs/TC-1018_10_1996.pdf.

[22] WfMC Workflow Standard Interoperability -- Wf-XML Binding, Document Number WFMC-TC-1023,http://www.wfmc.org/standards/docs/TC-1023_beta10.pdf.

[23] S. -B. Yan and F. -J. Wang, “A Cooperative Framework for Inter-Organizational Workflow System”, *Proceedings of the 27’th Annual International Computer Software and Applications Conference (COMPSAC’03)*, pp. 64-71, 1993.

[24] M. Kwak, D. Han, and J. Shim, “A Framework Supporting Dynamic Workflow Interoperation and Enterprise Application Integration”, *HICSS 35*, 2002

[25] E. Bertino, E. Ferrari, and V. Atluri, “A Flexible Model Supporting the Specification and Enforcement of Role-based Authorizations in Workflow Management Systems”, in *Proceedings of the 2’nd ACM Workshop on Role-Based Access Control*, 1997.

[26] M. H. Kang, J. N. Froscher, A. P. Sheth, and K. J. Kochut, “A Multilevel Secure Workflow Management System”, in *Proceedings of 11’th Conference on Advanced Information Systems Engineering*, 1999.

[27] M. H. Kang, B. J. Eppinger, and J. N. Froscher, “Tools to Support Secure Enterprise Computing”, in *Proceedings of the 15’th Annual Computer Security Application Conference*, 1999.

[28] M. H. Kang, J. S. Park, and J. N. Froscher, “Access Control Mechanisms for Inter-Organizational Workflow”, in *Proceeding of the 6’th ACM symposium on Access Control Methods and Technologies*, 2001.

[29] S. -C. Chou, A. -F. Liu, and C. -J. Wu, “Preventing Information Leakage within Workflows That Execute among Competing Organizations”, to appear in *Journal of Systems and Software*.

[30] http://www.wfmc.org.

[31] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts, Sixth Edition*, John Wiley & Sons, 2002

[32] R. T. Simon and M. E. Zurko, “Separation of Duty in Role-Based Environments”, *Proceedings of New Security Paradigms Workshops*, 1997.

[33] G. -J. Ahn and R. Sandhu, “Role-Based Authorization Constraints Specification”, *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 207-226, 2000.