Secure Role-Based Workflow Models

Savith Kandala and Ravi Sandhu

Savith Kandala	*Ravi Sandhu*
CygnaCom Solutions.	*SingleSignOn.Net and George Mason University*
(An Entrust Technologies Company)	*Dept. of Information and Software Engineering, MS 4A4*
7927 Jones Branch Drive (100 West)	*George Mason University, Fairfax, VA 22030*
McLean, VA 22102	*e-mail: sandhu@gmu.edu*
e-mail: skandala@cygnacom.com	*Home page: www.list.gmu.edu*

Abstract: In this paper we introduce a series of reference models for Secure Role-Based Workflow systems. We build our models over the well-known RBAC96 framework. The RBAC96 model supports the notion of abstract permissions. The nature of permissions is highly dependent upon the implementation details of the system, so we interpret the permissions for a Workflow system in terms of its components such as tasks, instances of the tasks and operations on them like execute, commit, abort etc. With this interpretation, we show that most of the components of RBAC96 still remain intact. The only components that change are the nature of permissions and their assignment to roles. The models are developed using the recently introduced four-layer OM-AM framework (comprising objective, model, architecture and mechanism layers). In this paper, we focus on the top two layers of OM-AM. We systematically describe our security objectives and construct our models to address these objectives. We also formally describe the models in terms of their components and their interactions. The main purpose for proposing these models is to articulate requirements for building Secure Role-Based Workflow Systems.

Key words: Access control, Role-Based workflows, workflow authorizations, and RBAC

1. INTRODUCTION

Workflow Management Systems (WFMS) are used to coordinate and streamline business process in an enterprise. A workflow defines various activities of an enterprise in terms of certain well-defined tasks. Users according to organizational rules carry out these tasks. Quite often, roles represent organizational agents intended to perform certain job functions within the organization. Users in turn are assigned appropriate roles based on their qualifications and responsibilities. [2]

RBAC96 [7] is a general model for role-based access control (RBAC). It treats permissions as un-interpreted symbols. The nature of permissions in the RBAC96 model is highly dependent upon the implementation details of a system and the general kind of system that it is. For example, an operating system protects files, directories, devices, ports, etc., with operations such as read, write, execute, etc., a relational database management system on the other hand protects relations, tuples, attributes, views, etc., with operations such as SELECT, UPDATE, DELETE, INSERT, etc. More generally, RBAC96 allows for abstract permissions specific to applications such as CREDIT and DEBIT in an accounting application. The nature of permissions in a WFMS can also be interpreted similarly, as a WFMS should control access to the tasks and instances of these tasks in the system with operations such as execute, commit, abort etc.

Preliminary ideas for Secure Role-Based Workflow models were presented in Transaction Control Expressions (TCEs) [6]. The TCE model is very natural and intuitive, and in fact reflects the world of forms and books in a Computer-Based System. However, TCEs were proposed much before RBAC96 was conceptualized and as such does not have all the components and specifications of RBAC96. The Task-Based Authorization Control (TBAC) [12] model was introduced to provide the notion of just-in-time permissions. It enables the granting, usage tracking and revoking of permissions to be automated and coordinated with the progression of various tasks. From a conceptual standpoint, TBAC focuses on the processing states and life cycle of authorizations and therefore cannot be directly compared to RBAC96. Bertino, Ferrari and Atluri (BFA) [2] have recently proposed a model for specifying and enforcing authorization constraints for WFMS. The model emphasizes on constraint specification and enforcement and as such does not encompass all the concepts of RBAC96.

The main contribution of this paper is that it shows, by aptly defining the nature of permissions RBAC96 can be extended to model Secure Role-Based Workflows. A consequential contribution is that it shows the OM-AM framework is a useful tool for modeling secure systems. The main purpose for proposing these models is to articulate the requirements for building secure role-based workflow systems.

The rest of the paper is organized as follows. Section 2 of the paper briefly describes the OM-AM framework, which was used to construct our Secure Role-Based Workflow models. Section 3 of the paper describes the

RBAC96 model. Section 4 of the paper introduces our first model for Secure Role-Based Workflows with very simple security objectives. Section 5, Section 6, Section 7 introduce the models for Secure Role-Based Workflows with progressively complex security objectives. Section 8 of the paper gives the future work to be done and the conclusion.

2. THE OM-AM FRAMEWORK FOR SECUIRTY ENGINEERING

In this section we briefly describe the four-layer OM-AM framework for security engineering, a detailed description can be found in [9]. The four layers are objective, model, architecture and mechanism surrounded by a sea of assurance, which permeates all layers (as shown in Figure 1). Objective and model are concerned with articulating what the security objectives and attendant trade-offs are, while architecture and mechanism address how to meet these requirements. In this paper we use the top two layers of this framework to formulate the security objectives and build our models.

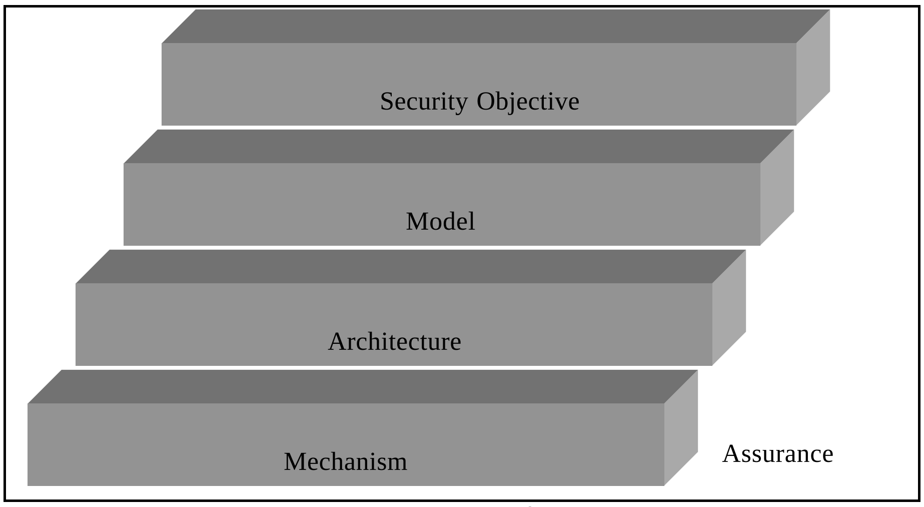

Figure 1 The OM-AM Framework for Security Engineering

3. THE RBAC96 MODEL

In this section we give a brief description of the RBAC96 model. This model has become a widely cited authoritative reference and is the basis of a standard currently under development by the National Institute of Standards and Technology. The main components of the RBAC96 model are users, sessions[1], roles, role hierarchy, permissions, user-assignment relationship, permission-assignment relationship and constraints. Figure 2 illustrates the RBAC96 model.

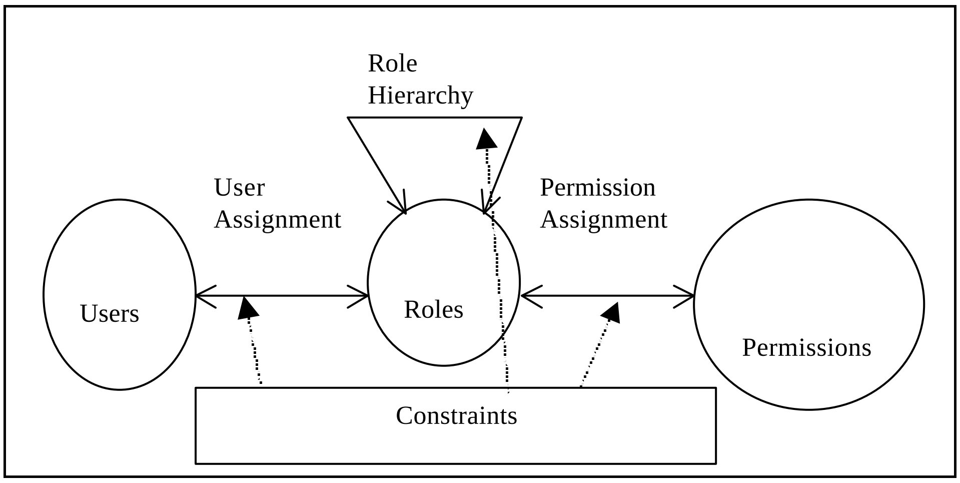

Figure 2 The RBAC96 Model

We now describe the RBAC96 model in terms of the OM-AM framework as described in section 2 of this paper.

Security Objective: The basic objective of RBAC is to simplify access control administration. It also provides ease of support for important security principles, particularly (1) least privilege, (2) separation of duties, (3) abstract permissions and (4) separation of administration and access.

Model

The RBAC96 model has the following components:

U – set of users, R – set of roles, P – set of permissions

$UA \subseteq U \times R$ (User Assignment)

$RH \subseteq R \times R$ is a partial order on R also called the role hierarchy or

Role dominance relation written as $\leq$

$PA \subseteq R \times P$ (Permission Assignment)

permissions: $R \to 2^P$, a function mapping each role r to a set of permissions.

permissions:* $R \to 2^P$ extends permissions in presence of a role-hierarchy.

permissions$(r_i) = \{ p \in P \mid (p, r_i) \in PA\}$

*permissions**$(r_i) = \{ p \in P \mid (\exists\, r \leq r_i)[(p, r) \in PA]\}$

Constraints are predicates, which applied to various components, determine if its value is acceptable or not.

[1] To simply our discussion we omit this component since it does not impact the results of this paper.

4. SECURE ROLE-BASED WORKFLOW MODEL$_0$

In this section we describe our first model for Secure Role-Based Workflows. We start by describing the various components of a Workflow System. A *task* in this model can be a program, a process or a procedure that is stored in the schema of the Workflow System. A *task instance*[2] is an instance of the task, or in other words it is a copy of the task that is made to run an instance of it. We also define an *Instance Mapping* that maps each task to its instances and is defined as follows.

Definition 1: Instance Mapping

Let TT be a set of tasks and TI a set of task instances, the Instance Mapping Á: TT ® 2^{TI} is a mapping that maps each task to its various instances, such that Á (a) Ç Á (b) = f if a ¹ b and a, b Î TT

We interpret *permission* in this model to be an authorization to *execute* a task. We further refine our interpretation by specifying that the permission to execute a task implies the permission to execute any instance of the task.

In order to model our interpretation of permissions, we define *Explicit Permissions* (EP) to be a cross product between the set of operations and the set of tasks. These Explicit Permissions are assigned to roles. We call this permission assignment relation as *Explicit Permission Assignment* (EPA).

We call permissions on task instances as *Implicit Permissions* (IP), and define them as a cross product between the set of operations and the set of task instances. For the moment we introduce a single operation on a task instance, called *execute* which authorizes execution of the task. Subsequent models will introduce additional operations. The permissions on task instances (IP) are assigned to roles based on EPA. If a task is assigned to a role then all instances of the task are also assigned to the same role. We call this assignment relation as Implicit Permission Assignment (IPA).

The crux of our Secure Role-Based Workflow models lies in these two relations EPA and IPA. Figure 3 below illustrates the model completely.

We now describe our model in terms of the OM-AM framework as described in section 2 of this paper.

Security Objective: Permissions in a Workflow System are interpreted as the authorization to execute tasks. Permission to execute a task implies permission to execute any instance of the task.

Model: U, R, RH, UA are unchanged from RBAC96

OP = {execute} (singleton set which contains the execute operation)

TT – set of tasks, TI – set of task instances

$\Im$ – An instance mapping that maps each task to its instances

[2] We consider the details like how these task instances are created, or how these task instances come into existence to be outside the scope of this paper. We leave these details to the Workflow Management System.

$\Im : TT \rightarrow 2^{TI}$ such that $\Im(a) \cap \Im(b) = \phi$ if $a \neq b$ and $a, b \in TT$
EP (set of EXPLICIT PERMISSIONS) = $OP \times TT$
IP (set of IMPLICIT PERMISSIONS) = $OP \times TI$
P (set of permissions) = $EP \cup IP$
EPA (set of explicitly assigned permissions) $\subseteq R \times EP$
IPA (set of implicitly assigned permissions derived from EPA)
IPA = $\{(r_i, \text{execute}, t_i) \mid [\exists (r_i, \text{execute}, t) \in EPA] \wedge t_i \in \Im(t)\}$
PA (Permission Assignment) = $EPA \cup IPA$
permissions: $R \rightarrow 2^{IP}$, a function mapping each role *r* to a set of permissions.
permissions:* $R \rightarrow 2^{IP}$ extends permissions in presence of a role-hierarchy.
permissions(r_i) = $\{(\text{execute}, t_i) \mid (\exists [(r_i, \text{execute}, t) \in EPA] \wedge t_i \in \Im(t)\}$
permissions(r_i)*=$\{(\text{execute}, t_i) \mid (\exists r \leq r_i)[(r, \text{execute}, t) \in EPA] \wedge t_i \in \Im(t)\}$

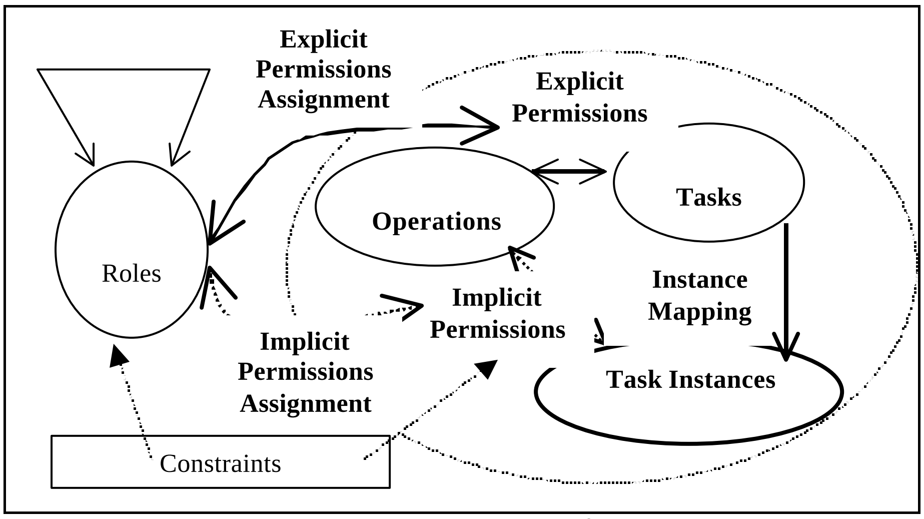

Figure 3 Secure Role Based Workflow Model $_0$

A proof-of-concept implementation to demonstrate the practical feasibility of this model is described in [1]. The system implemented ensures that users belonging to a specific role can execute the task instances. This model was not formally described in [1] we have described it formally here. The paper [1], describes an experiment to inject RBAC into an existing web-based workflow system using commercial-of-the-shelf (COTS) technology with minimal changes to the existing system.

5. SECURE ROLE-BASED WORKFLOW MODEL$_1$

In this section we describe our second model for Secure Role-Based Workflow systems. An obvious shortcoming of the previous model is that there is no notion of the task instances being completed. Therefore, there is no restriction on how many times a task instance can be repeatedly executed. We introduce the notion of states in tasks, and constrain the state transitions and the operations possible in each state in this model to improve upon the previous model.

For the purpose of this paper, we only consider transactional tasks. We believe that non-transactional tasks as well as two-phase commit tasks can also be modeled in a similar way. Figure 4 below, illustrates the states and the state transitions for some task structures.

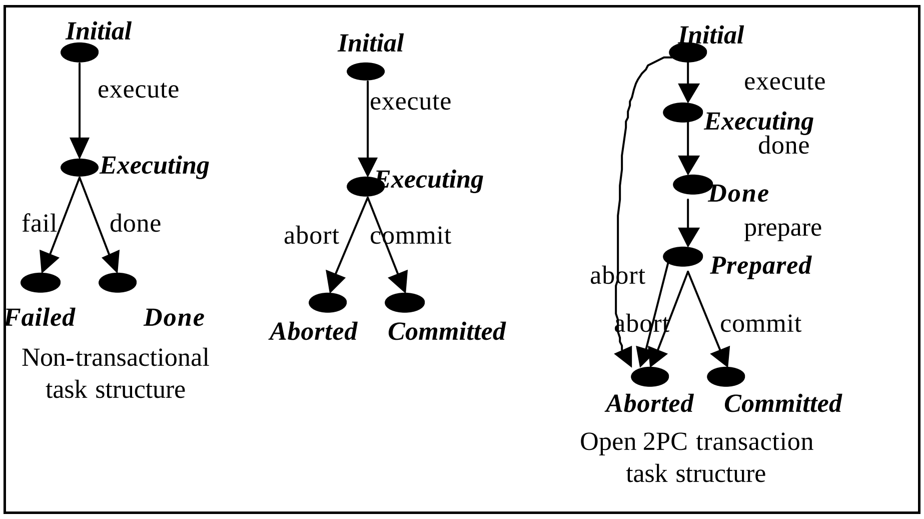

Figure 4 Some Task Structures

Consider the transactional task structure; the task is in the *Initial* state. The *execute* operation transitions the task instance to the *Executing* state. From this state there are two operations that are possible, *commit* and *abort*. The *commit* operation transitions the task instance to the *Committed* state and the *abort* operation transitions the task instance to the *Aborted* state. Although, we do not emphasize this in figure 3 below, from the *Aborted* state task instances are put back in the *Initial* state, so that they could be tried again for successful execution.

It should be noted that this task structure does not determine the means of execution or the functionality of the task, but only a high-level description of the (visible) state transitions.

We further restrict the permissions on task instances by stating that if a task instance is in the *Initial* state then the only possible operation on it should be *execute*. Similarly, if the task instance is in *Executing* state, then the possible operations should be *commit* or *abort*. By further restricting the assignment of implicit permissions we ensure that only those task instances that are in the proper state are assigned to roles. In order, to get the current state of a task instance and list possible operations of a task instance we define two functions *CurrentState* and *PossibleOperations*.

Definition 2: CurrentState determines the current state of the task.
Let TI be the set of task instances and S the set of states.
CurrentState: TI ® S maps each task instance to its state.
CurrentState(ti) = { s | s Î S and s is the current state of ti}

Definition 3: PossibleOperations determines the operations possible in a given state.
Let S the set of states and OP the set of operations
PossibleOperations $\subseteq$ *S X OP specifies the operations possible in each state. (For transactional tasks we consider PossibleOperations = {(Initial, execute), (Executing, commit), (Executing, abort)})*

As mentioned earlier, the crux of our Secure Role-Based Workflow models lies in the two relations EPA and IPA. The definitions for EP, IP and EPA remain unchanged from our previous model. The definition IPA is changed to include its dependency on EPA and the state of the task instance.

In order for the model to be complete we should consider the following constraints. First, either all permissions ((execute, t), (commit, t), (abort, t)) on the task t are assigned to a role or none of them are assigned. Second, the user who invokes the operation *execute* on the task instance, should also invoke either *commit* or *abort* operation on the same task instance. We list these constraints informally in the model.

We now describe our model in terms of the OM-AM framework as described in section 2 of this paper.

Security Objective: Security Objective for Role-Based Workflow Model$_0$ + Tasks have states (Initial, Executing, Committed, Aborted) and only certain operations (execute, commit, abort) can be performed in each state.

Model

U, R, RH, UA, TT, TI, EP, IP are unchanged from RBWM$_0$
OP (set of operations) = {execute, commit, abort}
S (set of states) = {Initial, Executing, Committed, Aborted}
T (set of state transitions) = {(Initial, execute, Executing), (Executing, commit, Committed), (Executing, abort, Aborted)}
CurrentState(ti) = { s | s $\in$ S and s is the current state of ti}
PossibleOperations = {(Initial, execute), (Executing, commit), (Executing, abort)}
P = EP $\cup$ IP
EPA $\subseteq R \times$ EP (Set of explicitly assigned permissions)
IPA = $\{(r_i, op, t_i) \mid [\exists (r_i, op, t) \in EPA] \wedge [t_i \in \Im(t)] \wedge$
$[(\mathit{CurrentState}(t_i), op) \in \mathit{PossibleOperations}]\}$
PA = EPA $\cup$ IPA

permissions: $R \rightarrow 2^{IP}$, a function mapping each role *r* to a set of permissions.
permissions:* $R \rightarrow 2^{IP}$ extends permissions in presence of a role-hierarchy.
permissions$(r_i) = \{(op, t_i) \mid [\exists\, (r_i, op, t) \in EPA] \wedge [t_i \in \Im(t)] \wedge$
$[(\mathit{CurrentState}(t_i), op) \in \mathit{PossibleOperations}]\ \}$
*permissions**$(r_i) = \{(op, t_i) \mid (\exists\, r \leq r_i)[\, (r, op, t) \in EPA] \wedge [t_i \in \Im(t)] \wedge$
$[(\mathit{CurrentState}(t_i), op) \in \mathit{PossibleOperations}]\ \}$

Constraints:

Either all permissions ((execute, t), (commit, t), (abort, t)) on the task t are assigned to a role or none of them are assigned.

The user who invokes the operation *execute* on the task instance, also is the only one who can invoke either *commit* or *abort* operation on the same task instance[3].

6. SECURE ROLE-BASED WORKFLOW MODEL$_2$

In this section we describe our third model for Secure Role-Based Workflows. We introduce the notion of task ordering at this stage. A workflow is considered to be a set of task types and their order of execution. Intertask dependencies determine how tasks in the workflow are coordinated for execution. The general type of dependency that is of interest is the *state dependency*. We use the Workflow Specification Language (WSFL) described in [4] to describe the workflow and the intertask dependencies. We focus on how the tasks can be placed together in a workflow with the use of state dependencies. The details of the language are not discussed in this paper, but can be found in [4].

A state dependency specifies how a transition of a task depends on the current states of other tasks. A state dependency is specified as a rule consisting *<left hand side> evaluator <right hand side>*. The state dependencies are expressed using the evaluator $\Rightarrow$, such that the left-hand side includes a predicate over task states and the right-hand side refers to a transition. For example, the following dependency specifies that if T1 is in a Committed state then T2 must transition to Initial state.

(T1, Committed) $\Rightarrow$ (T2, Initial);

Definition 5: (Workflow)

A workflow is defined as a compound task that is composed of a set of tasks and a set of intertask dependencies associated with it, which specifies the order of task execution. The intertask dependencies are specified in terms of state dependencies.

[3] The identity of the user who performs the operations should be recorded to enforce this constraint. This is a mechanism issue, which belongs at the bottom layer of OM-AM. As such it is out of scope for this paper where our focus is on the top two layers of OM-AM.

Examples

We discuss an example to demonstrate some of the specification features. The example has three tasks Initial Review, Correct Errors, and Process Application, and their order of execution is illustrated in figure 5 below.

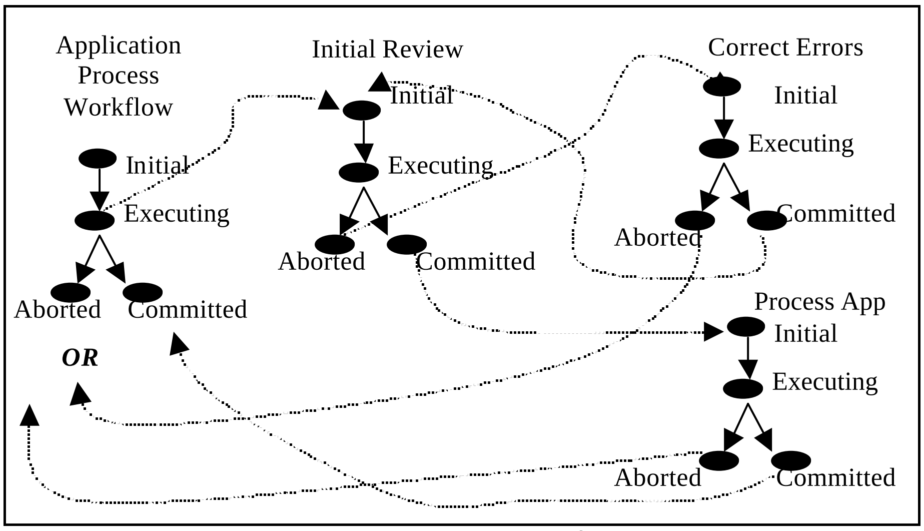

Figure 5 Application Process Workflow (example).

In terms of the above definition for Workflow we can write the intertask dependencies as follows:

Application Process Workflow = {Initial Review, Correct Errors, Process Application}

Intertask Dependencies:

(Application Process Workflow, Executing) ⇒ (Initial Review, Initial);
(Initial Review, Aborted) ⇒ (Correct Errors, Initial);
(Correct Errors, Committed) ⇒ (Initial Review, Initial);
(Initial Review, Committed) ⇒ (Process Application, Initial);
(Process Application, Committed)⇒(Application Process Workflow, Committed);
(Process Application, Aborted) ⇒ (Application Process Workflow, Aborted);
(Correct Errors, Aborted) ⇒ (Application Process Workflow, Aborted);

Intuitively, the intertask dependencies specified are consistent if and only if every path (with the exception of loops) from the Workflow *Initial* state terminates in the Workflow *Aborted* state or Workflow *Committed* state. Notice that the following scenario can occur Task *Initial Review* moves to aborted, Task *Correct Errors* is started and then committed, then Task *Initial Review* is moved to Initial, and so on. We assume that the loops eventually terminate and then the workflow terminates in either an aborted or committed state. We also consider formally specifying consistency checking of intertask dependencies to be outside of the scope of this paper and therefore do not specify them here. The *StartCondition Function* evaluates the intertask dependencies for the task instance and returns a boolean value TRUE or FALSE.

Definition 6: StartCondition Function evaluates the intertask dependencies for the task instance and returns a boolean value TRUE or FALSE.

Let TI be the set of task instances.

StartCondition: TI ® {TRUE, FALSE}.

As mentioned earlier, the crux of our Secure Role-Based Workflow models lies in the two relations EPA and IPA. The definitions of EP, IP and EPA remain unchanged from our previous model. The definition IPA is changed to include its dependency on EPA, the state of the task instance and the start condition of the task instance.

Definition 7: IPA – Implicit Permission Assignment.

IPA = {(r$_i$, op, t$_i$) | [$ (r$_i$, op, t) Î EPA] Ù [t$_i$ Î $\Im$(t)] Ù
[(CurrentState(t$_i$), op) Î PossibleOperations] Ù
[(StartCondition(t$_i$) = TRUE] }

We now describe our model in terms of the OM-AM framework as described in section 2 of this paper.

Security Objective: Security Objective for Role-Based Workflow Model$_1$ + Tasks are executed according to the specified intertask dependencies.

Model

U, R, RH, UA, TT, TI, OP, T, S, EP, IP, EPA, CurrentState, PossibleOperations, are unchanged from RBWM$_1$

StartCondition: TI $\rightarrow$ {TRUE, FALSE}

P (set of permissions) = EP $\cup$ IP

EPA (set of explicitly assigned permissions) $\subseteq$ R $\times$ EP

IPA (set of implicit permissions) =
{(r$_i$, op, t$_i$) | [$\exists$ (r$_i$, op, t) $\in$ EPA] $\wedge$ [t$_i$ $\in$ $\Im$(t)] $\wedge$
[(*CurrentState*(t$_i$), op) $\in$ *PossibleOperations*] $\wedge$
[(*StartCondition*(t$_i$) = TRUE] }

PA = EPA $\cup$ IPA

permissions: R $\rightarrow$ 2^{IP} , a function mapping each role *r* to a set of permissions.

permissions:* R $\rightarrow$ 2^{IP} extends permissions in presence of a role-hierarchy.

permissions(r$_i$) = {(op, t$_i$) | [$\exists$ (r$_i$, op, t) $\in$ EPA] $\wedge$ [t$_i$ $\in$ $\Im$(t)] $\wedge$
[(*CurrentState*(t$_i$), op) $\in$ *PossibleOperations*] $\wedge$
[(*StartCondition*(t$_i$) = TRUE] }

permissions(r$_i$)* = {(op, t$_i$) | ($\exists$ r $\leq$ r$_i$)[(r, op, t) $\in$ EPA] $\wedge$ [t$_i$ $\in$ $\Im$(t)] $\wedge$

[(*CurrentState*(t_i), op) ∈ *PossibleOperations*] ∧
[(*StartCondition*(t_i) = TRUE] }

Constraints:
Same as the constraints for Role-Based Workflow Model$_1$

7. SECURE ROLE-BASED WORKFLOW MODEL$_3$

In this section we describe our fourth and final model for Secure Role-Based Workflows. We introduce notion of specifying authorization constraints on tasks at this stage. We mainly focus on the Separation of Duties (SOD) constraints. We use the syntax specified in Transaction Control Expressions (TCEs) [6] to model these constraints. Extensions beyond this fourth model may be possible but are beyond the scope of this paper. In other words we are not claiming that this is the last word on this topic. We now briefly describe the syntax proposed in TCEs for expressing SOD constraints by an example. The details of the syntax are not discussed here, but can be found in [6]. Consider the workflow example Process Checks with three tasks Prepare, Approve, Issue. If all the three tasks are to be performed by different users, we do not associate any symbols with the tasks and simply write them as

(Prepare,), (Approve,), (Issue,)

If any user can perform task Approve, and the task Prepare and task Issue have to be performed by different users we write them as follows

(Prepare,), (Approve, ↑), (Issue,)

If the tasks Prepare and Issue have to be performed by same user and task Approve has to be performed by a different user we write them as follows

(Prepare, ↓x), (Approve,), (Issue, ↓x)

(Note: The token "x" is for relating multiple anchors. For instance we can use the symbol ↓ with a token "y" to identify another set of tasks that need to be performed by the same user.)

Definition 5: (Workflow)

A workflow is defined as a compound task that is composed of a set of tasks, a set of intertask dependencies associated with it, which specify the order of task execution and a set of TCE constraints. Each element in the set of TCE constraints is a two tuple (t, symbol) where t Î set of tasks and symbol is - *or* ¯ *(with token).*

In order to enforce these constraints, the identity of the user who performs each task should be recorded. The TCE constraints can be viewed as constraints on the *Permissions* component of RBAC96. We are basically checking if the user attempting to perform an operation on a task instance satisfies the TCE constraints for the workflow instance.

We now describe our model in terms of the OM-AM framework as described in section 2 of this paper.

Security Objective: Security Objective for Role-Based Workflow Model$_2$ + Separation of Duty and related Constraints

Model

The model is unchanged from Role-Based Workflow Model$_2$, except for the additional constraint on Implicit Permissions.

Constraints:

Same as the constraints for Role-Based Workflow Model$_2$ +
The user attempting to execute Implicit Permission should satisfy the TCE constraints.

8. CONCLUSION AND FUTURE WORK

We have presented a series of Secure Role-Based Workflow models, which systematically span the spectrum from very simple at one end to quite complex at the other. We started with a simple security objective and formulated our first Secure Role-Based Workflow model. For each subsequent model we added more complexity to the security objective and formulated our models by building on earlier ones. In this paper, we have shown that RBAC96 can be extended to model Secure Role-Based Workflows. The models were formulated following the OM-AM framework, thus demonstrating the effectiveness of the framework. The models developed realize the promise of RBAC96, which is a policy neutral, flexible, easy-to-customize framework for articulating and enforcing access control policies. Each of the models proposed can be implemented, depending upon the requirements of the system. For example, we have given a reference to the implementation of Model$_0$ [1], where the requirement was to just ensure that a user belonging to a specific role could execute a task.

The work presented in this paper can be extended along several directions. One such possible direction could be moving further down the OM-AM framework in terms of Architecture and Mechanisms to implement the models. Another possible direction could be extending these models for distributed and heterogeneous workflow systems. Yet another possible research direction could be to investigate the possibility of fitting our models into an existing Workflow System. Finally, incorporating issues related to delegation and controlled overriding of constraints into these models is also a challenging research goal.

ACKNOWLEDGEMENT

This work is partially supported by the National Science Foundation at George Mason University.

REFERENCES

[1] Ahn G, Sandhu R, Myong H K, Park J. (2000) Injecting RBAC to Secure a Web-based Workflow System. Fifth ACM Workshop on RBAC, pp 1 – 10
[2] Bertino, E., Ferrari, E. and Atluri, V. (1997). A flexible model for the specification and enforcement of authorization constraints in workflow management system. Proceedings of the Second ACM Workshop on Role-Based Access Control.
[3] Clark, D.D. and Wilson, D.R. (1987). A comparison of commercial and military security policies. Proceedings of IEEE Symposium on Security and Privacy, pp. 184-194.
[4] Narayanan K, Sheth A, (1995) Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operation. Distributed and Parallel Databases vol 3 number 2 April 1995 pp 155 - 186 [5] Nash, M.J. and Poland, K.R. (1987). Some conundrums concerning separation of duty. Proceedings of IEEE Symposium on Security and Privacy, pp. 201-207.
[6] Sandhu, R. (1988). Transaction control expressions for separation of duties. Proceedings of the Fourth Aerospace Computer Security Applications Conference, pp. 282-286.
[7] Sandhu, R., Coyne, E.J., Feinstein, H.L. and Youman, C.E. (1996). Role-based access control models. IEEE Computer, 29(2), pp. 38-47.
[8] Sandhu, R. (1990). Separation of duties in computerized information systems. Proceedings of the IFIP WG 11.3 Workshop on Database Security.
[9] Sandhu, R, (2000) Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way. Fifth ACM Workshop on RBAC, pp 111 – 119
[10] Sandhu R, Ferraiolo D, Kuhn R, (2000) The NIST Model for Role-based Access Control: Towards a Unified Standard, Fifth ACM Workshop on RBAC, pp 47-64
[11] Simon, R.T. and Zurko, M.E. (1997). Separation of duty in role-based environments. Proceedings of Computer Foundations Workshop X.
[12] Thomas, R.K. and Sandhu R. (1997). Task-based authorization controls (TBAC) Proceedings of the IFIP WG 11.3 Workshop on Database Security.
[13] Ullman, J (1989) Principles of Database and Knowledge-Base Systems (2nd volume). Computer Science Press, New York.