Formal Verification of Business Workflows and Role Based Access Control Systems

Arnaud Dury[1], Sergiy Boroday[1], Alexandre Petrenko[1], Volkmar Lotz [2]
[1] *Computer Research Institute of Montreal (CRIM),* [2] *SAP Labs France*
FirstName.LastName@crim.ca, FirstName.LastName@sap.com

Abstract

An approach for combined modeling of role-based access control systems (RBAC) together with business workflows is presented. The model allows to model check various security properties. Several techniques to confine the state explosion, which may occur during model checking are presented and experimentally evaluated using the model checker Spin. The techniques allow the verification of the business workflow and associated RBAC for a reasonable number of users of a medium sized company.

1. Introduction

Role Based Access Control (RBAC) is an increasingly popular and efficient security solution. A major advantage of RBAC is its ability to constraint malicious or erroneous user behavior, typically using the concept of separation of duties (SoD), allowing several persons to complete a critical task without anyone having excessive control. Currently, RBAC design and maintenance of RBAC security policies are challenging problems as company structure, roles, user pools, business workflows, internal and external (legal) security requirements are always changing. In this context, automated RBAC verification techniques can contribute both to product integrity and time to market. While early RBAC verification methodologies rely on visualization of constraints [3] and graph transformation [4], modern, more formal and powerful approaches are usually based on automated reasoning techniques, such as model checking.

In this work, we aim at model checking of RBAC in the context of workflows of business applications. We first present a short overview of previously developed approaches to RBAC and workflow verification. We then describe our approach that is based on model checking of business workflow considered together with a RBAC. Since a major issue with model checking is usually state explosion, we pay a special attention to simplification (abstraction) methods. We model RBAC in conjunction with workflow processes, in a setting derived from a real application context. We check compatibility of RBAC with a given workflow and validate security properties against the given RBAC constraints set and workflow. Our work differs from the "light-weight" set-theoretic model checking efforts [5], [6], [7], [8], [9] by considering the workflow on which the RBAC is imposed and order-dependent constraints. Unlike most work on full-scale model checking with Spin [10], [11], [12], [13], we elaborate several techniques which fight the state explosion problem (we refer to our technical report for more details [1]).

The paper is organized as follows. In Section 2, we provide a short overview of existing results related to RBAC and workflow verification. In Section 3, an approach for modeling RBAC together with business workflows using Extended FSM (EFSM) is presented. Section 4 describes a case study of Procure to Stock Workflow. In Section 5, we consider various techniques, which alleviate state explosion in model checking RBAC on a given workflow. In Section 6, we provide the results of experimental evaluation and comparison of the proposed techniques.

2. Related Work

2.1. Workflow Model Checking

Workflows reflect organizational aspects of a work procedure, such as structure, synchronization and ordering (flow) of tasks, information flow, etc. Workflow notation and languages, such as EPC/ARIS [28] are supported in business software, notably by SAP. Security properties of business applications result not only from access control mechanisms, but also from business workflow implementations. For example, if the authorization of a purchase request may precede (in all or some executions) its actual filing in a given

0-7695-2989-5/07 $25.00 © 2007 IEEE
DOI 10.1109/SECURWARE.2007.30

workflow, then the purchase request can be approved with some field left blank (violation of a so-called "no carte blanche" security property). Usually workflow modeling is based on graphs, automata, Kripke structures, Petri Nets, and more rarely on constraint solving, data-flow pointer analysis, BDD, propositional and temporal logic [14].

However, despite active research and development, available workflows verification tools are not numerous and do not always support data [15] or cycles [16]. A great deal of research is devoted to workflow model checking related to web service verification [17]. Among available specialized web service/workflow model checking tools we could mention WSAT tool [18], that supports several web service languages, including BPEL4WS and could use both Spin and NuSMV as a backend. Since suspension of workflows is often difficult and undesirable, the problem of dependable introduction of policy changes without interrupting currently executed workflows arises [19]. The proposed solution is based on Spin model checking of workflows against properties such as "jobs are billed if and only if they are shipped" [19].

Thus, there exist supporting tools that translate workflow models into input languages of model checking tools (usually automata or Petri net based), so in this work we assume an automata model of a workflow is already available.

2.2. RBAC Model Checking

Even a relatively simple case of verification of static SoD policies (such as "at least n user are required to perform a given task") from static role mutual exclusion constrains (such as a given user cannot be assigned to more than a certain number of roles from a given set of allowed roles) is computationally difficult (coNP-complete) [20]. Thus, "heavy artillery" of model checking that involves sophisticated optimization techniques to cope with hard problems (SAT-solvers, BDD, partial order reduction etc…) is justified for RBAC verification. Most of known work apply model checking to RBAC policies, abstracting from implementation details and control mechanisms.

In [11], authors verify application and organization specific RBAC "policy implementations" against selected security properties that represent a high level enterprise security policy. Security policies may be of a higher level and evolve independently of their RBAC implementations. Among verified properties are static, dynamic (for one or all sessions), operational, object, history based SoD, prerequisite, cardinality, and user-user[1] constraints. Some of the verified history based properties appear to be more relevant to business logic than to security (e.g., a recorded invoice is eventually verified). The RBAC implementation model consists of four concurrent processes: the first process selects user, the second selects a role, the third – a permission, and the fourth verifies relevant constraints, and, if no constraint is violated, models the operation, associated with the permissions by recording the user id into a designated "history" array. While history array does not scale well, alternative approaches, such as using a blacklist [21] of operations for each user or enabled and disabled states for each permission assigned to a role are proposed.

Ahmed and Tripathi [10] verify Computer Supported Cooperative Work System, CSCW, using the model checker Spin against various security properties. While manual specification of CSCW security properties in LTL might be difficult, "conversion functions" that facilitate the translation of SoD constraints to LTL are developed. Property specific abstraction is used to fight state space explosion. Four different Spin models are developed to verify four different aspects, namely, task flow (e.g., each operation could be executed), role constraints (e.g., each role could have a member), information flow (e.g., non inference), and administrative role assignment. To cope with state explosion, following measures are suggested: some operations could be excluded; verification based on a specific user often can be generalized to verification of global properties (user symmetry); abstraction of internal data structure (for example, in many cases, the model does not need to maintain the count of the events in the precondition that contributes to a property; instead, a bit signifying that the precondition is satisfied is maintained); for a faster verification, role constraints could be specified in LTL.

Specification language cTLA, derived from Lamport's Temporal Logic of Action (TLA), is used to formalize and after translation to Promela to verify RBAC [12]. Spin and temporal logic could be used for goal-elaborating policy refinement of a higher level policy to a low level policy [22]. Nguyen and Rathke [13] use Spin to verify multithreaded functional programs (rather than workflows or management systems) against security policies, expressed in form of a "policy automaton" which, generally speaking, could be used to express RBAC policies [23].

[1] A user-user constraint usually prevents two users from activating or being assigned the same role

A popular open-source model checker NuSMV is used to solve so-called safety analysis problem (whether only trusted users could violate a given security property) [24]. Furthermore, a more general Administrative Insider Threat Assessment Problem (AITAP) is formulated and verified. Access control schemes are formally defined as state-transition systems (i.e., labeled Kripke structures). The NuSMV model of RBAC is straightforward, except for an efficient model abstraction, based on pruning of irrelevant rules and role activations. In the reported case studies, the model checking approach is compared with the logic–programming approach, based on a Prolog like language. Experimental data favor the model checking approach. While state space explosion persists, RBACs with tens of roles and rules are verified.

A popular RBAC analysis tool is Alloy Analyzer, a model checker that supports a light-weight structural specification language Alloy, based on the first order logic [6], [7], [5], [8]. While language features are limited, and modeling of complex history based properties or workflows is difficult, it is perfectly appropriate for describing organizational structures and simple SoD constraints. A designated policy verification model checker RW is developed [25], [31]. Besides model checking, RBAC schema/policy verification could be performed using theorem proving [32], graphs, binary decision diagrams (BDD) [33], constraints solvers, and integer programming.

Summarizing the above, we notice that there are few reports on applying the model checking technology for verifying RBAC on workflows. Even when a workflow, on which a RBAC is imposed, is somehow modeled, technical details and experimental data are rarely reported. Here we try to elaborate this idea in detail. More specifically, we suggest an approach for verifying a set of properties (typically related to separation of duties and reachability concerns) over a Business Workflow and a given RBAC, using the model of Extended Finite State Machine (EFSM), which is often used in input languages of model checking tools. EFSM includes a set of states, actions, and guarded transitions along with optional variables and parameters. Several techniques alleviating state explosion caused by a growing number of users are experimentally evaluated.

3. An Approach for Modeling RBAC with Business Workflows

We formalize in this section the notions of roles, rules and properties. Then, the EFSM model of RBAC and Business Workflow is proposed.

3.1. Roles, Rules, and Properties of RBAC

The central notion of RBAC is the notion of role. Each role is associated with a non-empty set of user actions. In a formal set-theoretic RBAC definition, permission to role assignment relation is usually defined [30]. Every user action of the Business Workflow is associated to at least one role, though one user action can be associated to several roles at the same time. A user to role assignment defines which roles each user could activate during the execution of Business Workflow. Role activation can be allowed either in any state of the workflow, in specific states of the workflow (such as only in the initial state for instance), or can be constrained by a more elaborate restriction. Between those two extremes (activation in any or only in initial state), one meaningful role activation mode is to allow activation of a role only in Business Workflow states that have an emanating transition, labeled by an action of the role. This activation mode is assumed in this work from this point on. A user can activate several roles at the same time, but no deactivation is allowed.

Additional constraints could restrict role activation and assignment. We distinguish two types of RBAC constraints: dynamic and static. Static ones are imposed on the actions of the administrator, who assigns roles to users. Dynamic constraints are imposed on role activation during Business Workflow execution. Thus, dynamic ones could be in conflict with a user to role assignment. Since static constraints are well supported by existing tools, we focus on dynamic constraints. Our approach is related to rule-based RBAC systems, such as OASIS (Open Architecture for Securely Interworking Services) [29] in which role activation is governed a set of preconditions Contrary to OASIS, we support negative preconditions (such as "*Role* 1 is not active"), which are needed to implement separation of duties constraints. Conditions that are not based on roles (e.g., temporal or user specific conditions) are not considered in this work. The approach targets state reachability and simple separation of duties properties. While various preconditions could occur in rules of the RBAC, we consider only role based, i.e., related to previous activation (or non-activation) of other roles or propositional logic formulas that use negation, conjunction, or disjunction over atomic propositions denoting activity of certain roles.

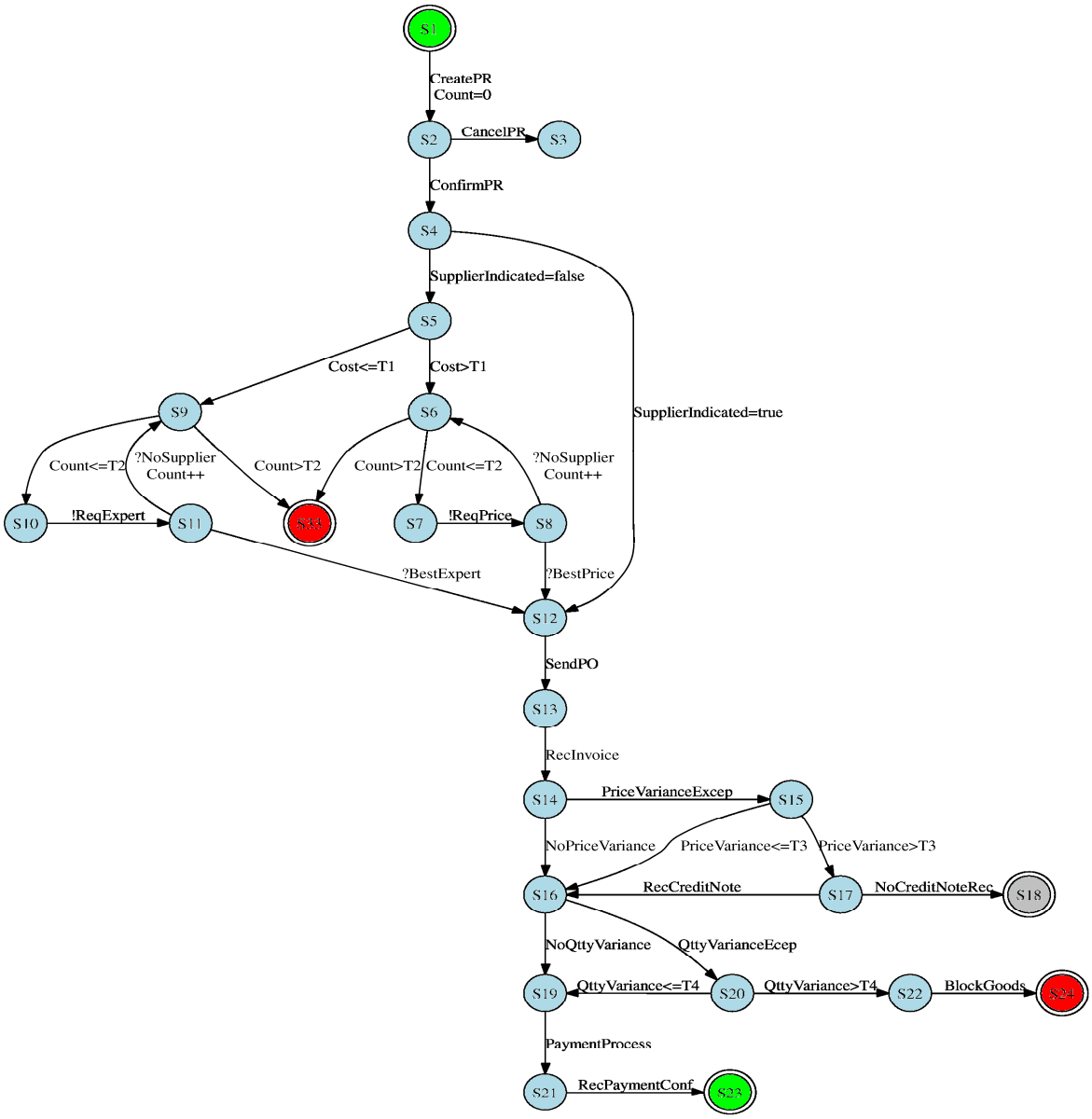

Figure 1 Procure to Stock Business Workflow

3.2. EFSM Modeling of RBAC on Workflow

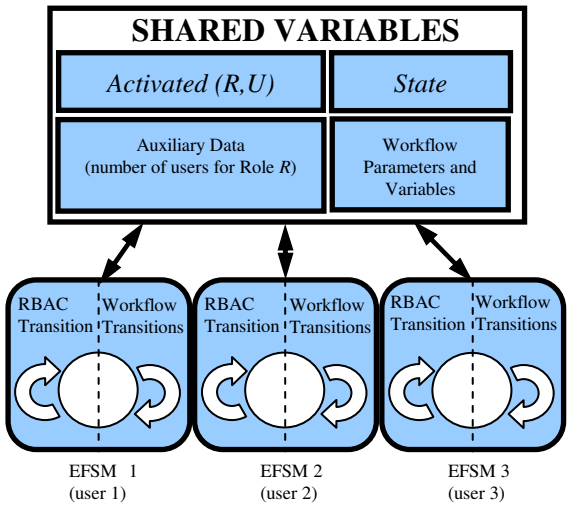

Figure 2 Model Overview

We considered at first three alternative EFSM models: a first one in which RBAC related variables, guards and transitions are added into the Business Workflow EFSM; a second one in which three separate EFSM are defined for, respectively, the Business Workflow, the non-deterministic choice of a user to execute an action, and the roles activations; and a third one using one single state EFSM per user. The last one was chosen for the reasons of compactness of the representation (role activation transition appears only once in the model specification), reduction of the number of explored states (no state is explicitly created when a user unable to perform any action is selected), and simplicity of implementation in the Spin model checker. Further details are given in [1]. In the chosen approach, each state of the Business Workflow is represented by a corresponding value of a designated variable shared among the users' EFSMs (see Figure 2). The transformation from the original Business Workflow EFSM to the single state Business Workflow EFSM is first explained, and then the derivation of the users' EFSM from such single state EFSM is presented.

A variable *State* is defined and contains the identifier of the current state of the Business Workflow. Each transition of the Business Workflow from state s_i into state s_j with the guard g and action a is represented by a loop of the single state EFSM with the guard g and *State* = s_i and action a' which combines the action a and assignment *State* = s_j.

The user EFSM is derived from the single state Business Workflow EFSM described above. For each existing role R that can be assigned to the user U we define a corresponding Boolean *Activated*(R, U), which is true when the role R is currently activated for the user U and false otherwise. For a given role and a state, in which the role can be activated a new looping transition is created in the EFSM. The action of the transition represents the activation of the role, while the guard is a predicate expressing the RBAC rules defined for this role. As an example, consider the following RBAC rule: "A user U can only activate *Role* 5 if and only if he previously did not activate *Role* 3 and *Role* 4". For such rule a new looping transition would be created in the EFSM. The guard of the transition would be the predicate "If *Activated*(3,U) = false and *Activated*(4,U) = false" and the action would be "*Activated*(5,U) = true". We represent a mutual exclusion of roles (SoD constraint) by the two following rules: A user U can only activate *Role* 2 if and only if he did not activate *Role* 1, and a user U can only activate *Role* 1 if and only if he did not activate *Role* 2. Modeling of role cardinality and user cardinality constraints is detailed in [1]. The choice of the user to execute a next action of the Business Workflow is performed by the model checker. The variable *State*, shared by all EFSMs (as well as workflow variables and parameters) enforces EFSM synchronization.

4. The Procure to stock workflow and RBAC

4.1. Procure to Stock Business Workflow

The Business Workflow used as a running example is called "Procure to Stock" and describes the procurement of goods from the creation of the purchase request, up to the delivery of the goods or the termination of the workflow in one of several unsuccessful end states. Figure 1 shows the Business Workflow EFSM. Message receives are used to model exceptions during the execution of the Workflow. For instance, there is an exception called *PriceVarianceException*, which occurs whenever the price received on the invoice differs from the price that was settled between the buyer and the seller, when the purchase order was placed. User actions are *CreatePR*, *ConfirmPR*, *CancelPR*, ?*NoSupplier* ("?" indicates a

message reception), !*ReqPrice* ("!" indicates a message sending), !*ReqExpert*, ?*BestPrice*, ?*BestExpert*, *SendPO*, *RecInvoice*, *RecCreditNote*, *NoCreditNoteRec*, *PaymentProcess*, *RecPaymentConf*, *BlockGoods*. The only variable is *Count*. Business Workflow posesses integer parameters *Cost*, *T1*, *T2*, *T3*, *T4* and the Boolean parameter *SupplierIndicated*. System actions, which are not attributed to any user, are: counter increments, ?*PriceVarianceException*, and ?*QuantityVarianceException*. The latter (exception message receives) are associated with input parameters *PriceVariance* and *QuantityVariance* (*QttyVariance* in the Figure 1), respectively. In the next section, a role set and role assignment for this Business Workflow are presented.

4.2. RBAC Rules

Actions are grouped in five roles, *Role* 1 through *Role* 5. The only action of *Role* 1 is *CreatePR*. The actions of *Role* 2 are *ConfirmPR*, *CancelPR*. The actions of *Role* 3 are !*ReqPrice*, !*ReqExpert*, ?*BestPrice*, ?*BestExpert*, ?*NoSupplier*, *SendPO*. The actions of *Role* 4 are *RecInvoice*, *RecCreditNote*, *NoCreditNoteRec*. The actions of *Role* 5 are *PaymentProcess*, *RecPaymentConf*, *BlockGoods*.

The following users are defined (in a small company): Alice (the CEO) can activate all the roles, Bob (the Supervisor) can activate *Role* 1, *Role* 2, *Role* 3 and *Role* 5, Carol (the Financial Manager) can activate *Role* 4 and *Role* 5. All the other users are Employees and can only activate *Role* 1 and *Role* 4.

In addition, we define two RBACs: a minimal one with only one rule and a more complex one with four rules restricting user/roles assignments. The simple RBAC, hereafter RBAC1, contains the only rule: *Role* 5 cannot be activated after *Role* 1. The second RBAC, hereafter RBAC2, contains the four rules: *Role* 2 cannot be activated after *Role* 1, *Role* 3 cannot be activated after *Role* 2, *Role* 5 cannot be activated after *Role* 3, and *Role* 5 cannot be activated after *Role* 4. For simplicity, we assume that the user pool contains one or more Employee(s), but CEO, Supervisor, and Financial Manager are single users.

4.3. Variables and Parameters Abstraction

In order to make the state space of the Business Workflow finite, we abstract (conservatively) the unbounded parameters: the integer parameters cost *T1*, *T2*, *T3*, *T4*, as well as the input parameters *QuantityVariance* and *PriceVariance*. The abstraction of these variables introduces a non-deterministic choice in the user EFSM. We delete the counter variable, and such deletion adds a cycle absent in the original Business Workflow (this is the only added path in the EFSM of the abstracted Business Workflow which is non-executable in the original Business Workflow). The added cycle does not prevent though a model checker from verifying any safety properties on actions or state reachability.

5. Alleviating State Explosion

We consider five techniques, which alleviate state explosion in model checking RBAC on a given workflow. They are related to data encoding (an efficient data representation), property reformulation, RBAC restriction, users' activities restriction, and user symmetry. We have chosen SPIN as our model checker of reference due to its general robustness and performance.

5.1. Data Encoding

An economical representation of data reduces memory consumption in model checking. The role activation array is the most memory-consuming variable of the user EFSMs and thus, the primary target for memory reduction. Unfortunately, the Promela Boolean variables require eight bits of memory, instead of only one. Thus, we apply a bit coding technique [2] representing eight Boolean variables by a single byte value. The technique does not lead to any reduction in the size of the explored state space, though it reduces the amount of memory needed to explore a given state space. The technique is property and RBAC independent.

5.2. Property Reformulation

Given a specific Business Workflow, some LTL properties can be replaced with simpler properties equivalent for the particular Business Workflow. As an example, a simple SoD property forbidding a user u to perform two actions *a* and *b* is stated in LTL as follows:

$$\Diamond\, u(a) \rightarrow !\, \Diamond\, u(b)$$

For a Business Workflow, which forbids any occurrence of *a* before *b*, it could be replaced with:

$$\Box(u(a) \rightarrow !\, \Diamond\, u(b))$$

The first formula rejects any execution containing either the sequence *a…b* or *b…a*, while the second only matches the sequence *a…b* and will be verified faster. The same approach can be applied for a set of users by building a LTL formula which is an enumeration of such single-user formulae. The approach requires model checking or static analysis of the Business Workflow in order to determine if *a* can occur before *b*, which, however, is less computationally expensive than the verification of RBAC along with Business Workflow. Such technique is relevant to separation of duties properties, but not to reachability properties.

5.3. RBAC Restriction

The idea is close to role pruning applied in the context of insider threat assessment in [24]. It consists in forbidding some activation of certain roles in given states. We consider a Business Workflow, in which there are no interleaving roles, i.e., on each path, between two actions of the same role, no other action of another role can be executed. We also assume that all roles in preconditions are negated. Then, activation of a given role could be restricted to a set of states, where each state is the first state of a path of the Business Workflow, whose emanating transition is labeled by an action of this role. For our running example, activation of *Role* 4 could be restricted to state *s*13 for instance. However, in a more general case, the Business Workflow may contain several possible paths on which a role can be activated first in different states of the Business Workflow. We, thus, restrict the activation of such a role to several states. For instance, *Role* 3 can be restricted to states *s*7 and *s*10, because on any possible execution, where actions of *Role* 3 are executed, the first states of those executions are *s*7 and *s*10. The case of interleaving roles is discussed in more details in [1].

5.4. Users' Activities Restriction

The idea of restricting users' activities is to impose additional constraints to disallow role activations and actions of specific users that are not relevant to a property to be checked. Restrictions exclude from exploration global states of the Business Workflow and RBAC model where the property cannot be violated. The idea is applicable at the least to the following case:

Property could be formulated in terms of actions of a single user, i.e., the first order LTL [26] formalization of the property contains at most one quantification over users (for instance, a simple SoD property), and each action of the SoD property cannot be executed more than once in any given execution.

Then we introduce a new constraint that prevents a user unable to execute all mutually exclusive actions of the SoD property (e.g., due to role assignment) from actually performing any of them. The reduction of the number of possible actions for a user alleviates the effects of state explosion. Furthermore, if all actions of a given role *R* for a given user cannot be executed due to the newly introduced constraints, we then introduce another constraint that prevents the user to even activate *R*, further reducing state explosion. For our SoD property example, we introduce the following two constraints: User 3 (Financial Manager) is prevented from executing action *ReceivePaymentConf* of *Role* 5, and Employees are prevented from executing action *CreatePR* of *Role* 1.

5.5 User Symmetry

Several Employees can take part in the execution of the workflow. Each of them is given the same set of roles, so should one of them be able to violate the SoD property then any of them would. Such user symmetry can be exploited to simplify the expression of the SoD property: instead of enumerating the SoD violation over every Employee taking part in a given execution of the workflow, the SoD violation is expressed for one arbitrary chosen Employee, though modifying the number of users does not look to be compatible with cardinality rules.

6. Experiments

In the experiments the number of users is increased until the verification can no longer be performed within the given memory. The Separation of Duty property to be verified is: "No user should be able to perform, in any given execution, both *CreatePR* and *RecPaymentConf*". The following results show the effect of state space explosion on the original model (no technique applied) and on the same model with one of the five techniques.

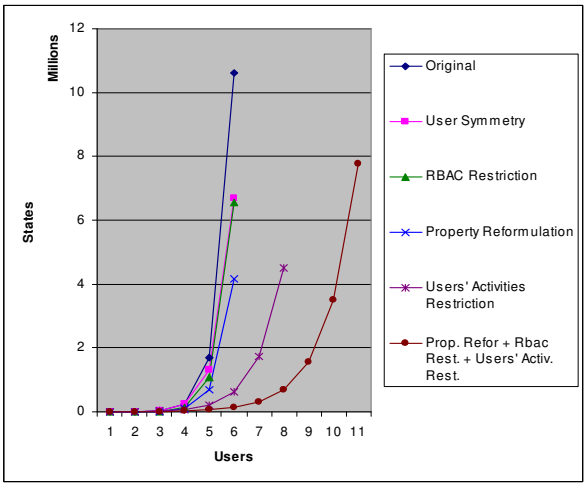

Figure 3 Results for RBAC 1

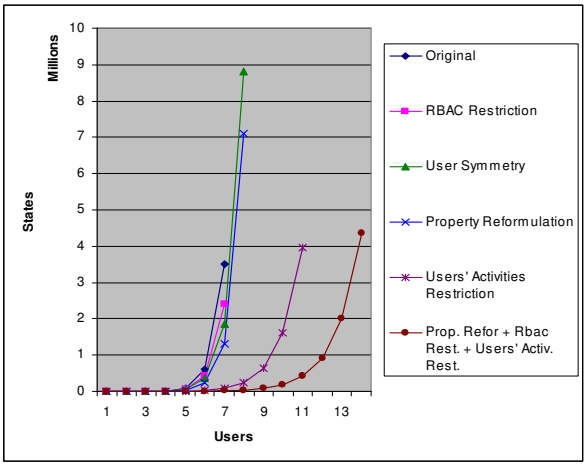

Figure 4 Results for RBAC 2

As explained before, the data encoding technique can safely be used in any case and does not affect the state space, and thus is used by default in all the reported experiments. All experiments are performed using Spin model checker, version 4.2.3 under Windows XP, on a computer with 2Gb of RAM, of which 1Gb is allocated to Spin. Results are shown in Figures 3 and 4.

7. Discussion and Future Work

Experiments indicate the proposed modeling approach using an EFSM for representing RBAC together with a Business Workflow is flexible enough to be combined with various techniques which can confine the state explosion during model checking triggered by a growing number of users. The experimental data show that among analyzed techniques, the most efficient one is the users' activities restriction; it reduces by up to 97.33% the number of explored states. In fact, all the techniques can be applied together, and our experiments show that the number of considered users nearly doubles in this case (from 6 to 11 for RBAC1, and 7 to 14 for RBAC2). Experiments with user symmetry indicates that this technique is less efficient than most of the other techniques. Additional experiments indicated that symmetry reduction combined with any of the other suggested techniques yields additional state reduction. However, when we performed an experiment combining all techniques except symmetry reduction, the additional inclusion of symmetry reduction did not further reduce the number of explored states Nevertheless, symmetry reduction is relatively easy to implement and combines with any of discussed techniques. Overall, the experimental data indicate that developing techniques for model reduction specific to the RBAC and Business Workflow is a research activity which merits further efforts.

Our ongoing work consists in developing another technique, which consists of pruning Business Workflow with respect to the property to validate. In order to prune some transitions and states, the possible occurrence of actions referred to in the property would be tested in every existing path of the model, but without taking into account any users. This could be performed with much less stringent memory requirements than that of the Business Workflow with the complete user pool. The approach would be simpler than CEGAR [27], since no counterexample would be used to refine a model, and model checking of an abstract property on an abstract model (Business Workflow) would lead to a straightforward simplification rather than to a refinement of Business Workflow.

Another aspect of our future work will be the study of interactions of various techniques, and the range of Business Workflows, rules, and properties, on which they can be applied. Such work could eventually form the basis of a library of useful security properties and patterns, facilitating property specification.

8. Conclusion

The paper contributes to a relatively unexplored domain of *combined* Business Workflow and RBAC analysis, namely, we propose a novel EFSM model for representing a Business Workflow with a rule based RBAC, suitable for formal verification (model checking). Moreover, in the context of EFSM based RBAC model checking, we studied five techniques to alleviate the state space explosion: data encoding, property reformulation, RBAC restriction, users'

activities restriction, and user symmetry. Two of these techniques (users' activities restriction and property reformulation) are new to the best of our knowledge and are the most efficient. We claim an improvement of up to 97.33% in terms of the number of explored states, and doubling the number of the users which can be assigned to execute a business workflow in presence of RBAC.

Experiments are performed using the model checker Spin on a simplified Business Workflow, which nevertheless possesses important real-life features, such as presence of parameters, messages exchanges, branching, and cycles. We, thus, believe that the obtained results are relevant to a useful range of more realistic Business Workflows. While we experimented only with a couple of RBAC constraints (rules), an encouraging observation is that a larger set of constraints is easier to model check, as it imposes more restrictions on possible user activities, and, thus, reduces the state space.

References

[1] A. Dury, S. Boroday, A. Petrenko, V. Lotz, *Model Checking Access Control in Business Workflow,* Technical Report CRIM-06/10-11, CRIM, Montreal, November 2006.

[2] T. Ruys, "Low-Fat Recipes for Spin", *SPIN Model Checking and Software Verification: Proceedings of the 7th International SPIN Workshop*, Springer Verlag, 2000.

[3] J.E. Tidswell, T. Jaeger, "An access control model for simplifying constraint expression", *7th ACM conference on Computer and Communications Security*, 2000.

[4] M. Koch, F. Parisi-Presicce, "Visual specifications of policies and their verification", *Workshop on Fundamental Approaches to Software Engineering*, LNCS 2621, 2003.

[5] A. Schaad, J. Moffett, "A lightweight approach to specification and analysis of role-based access control extensions", *7th ACM Symposium on Access Control Models and Technologies*, 2002.

[6] J. Zao, H. Wee. J .Chu, D. Jackson, "RBAC schema verification using lightweight formal model and constraint analysis", *ACM Symposium on Access Control Models and Technologies*, 2003.

[7] M. Mankai, L. Logrippo, "Access control policies: modeling and validation", *Notere, Conférence sur les Nouvelles technologies de la répartition, Canada*, 2005.

[8] G. Hughes, T. Bultan, *Automated verification of access control policies*, Technical Report 2004-22, University of California, Santa Barbara, 2004.

[9] S. Park, G. Kwon, "Verification of UML-based security policy model", *International Conference on Computational Science and its Applications*, 2005.

[10] T. Ahmed, A.R. Tripathi, "Static verification of security requirements in role based systems", *8th ACM Symposium on Access Control Models and Technologies*, 2003.

[11] F. Hansen, V. Oleshchuk, "Conformance checking of RBAC policy and its implementation", *The First Information Security Practice and Experience Conference*, LNCS 3439, 2005.

[12] P. Herrmann, "Formal security policy verification of distributed component-structured software", *23rd IFIP Intl. Conf. on Formal Techniques for Networked and Distributed Systems*, LNCS 2767, 2003.

[13] N. Nguyen, J. Rathke, "Typed static analysis for concurrent policy-based resource access control", Unpublished draft, http://www.cogs. susx.ac.uk/users/ julianr/pubs/dist-access.html, 2005.

[14] J. Wainer, F. de Lima Bezerra, P. Barthelmess, "Tucupi: a flexible workflow system based on overridable constraints", *ACM SIG Symposium on Applied Computing*, 2004.

[15] Woflan tool (on-line) http://is.tm.tue.nl/research/woflan/

[16] W. Janssen, R. Mateescu, S. Mauw, P. Fennema, P. Stappen, "Model checking for managers", SPIN 1999.

[17] M. Koshkina, F. van Breugel, *Verification of business processes for web services*, Technical Report CS-2003-11, York University Department of Computer Science, 2003.

[18] WSAT Tool: http://www.cs.ucsb.edu/~su/WSAT/

[19] P.K. Bose, M.G. Matthews, "Dynamic change in workflow-based coordination of distributed services", *International Workshop on Self-Adaptive Software*, 2001.

[20] N. Li, Z. Bizri, M.V. Tripunitara, "On mutually-exclusive roles and separation of duty", *ACM Conference on Computer and Communications Security,* 2004.

[21] J. Crampton, "Specifying and enforcing constraints in role-based access control", *8th ACM Symposium on Access Control Models and Technologies*, 2003.

[22] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, "A functional solution for goal-oriented policy refinement", *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, 2006.

[23] M. Covington, M. Moyer, M. Ahamad, "Generalized role-based access control for securing future applications", *23rd National Information Systems Security Conference*, 2000.

[24] S. Jha, N. Li, M. Tripunitara, Q. Wang, W. Winsborough, *Security analysis and administrative insider threat assessment in role-based access control*, CERIAS Tech Report 2005-77, Center for Education and Research in Information Assurance and Security, Purdue University, 2005.

[25] N. Zhang, M. Ryan, D.P. Guelev, "Synthesising verified access control systems in XACML", *ACM Workshop on Formal Methods in Security Engineering*, 2004.

[26] E.A. Emerson, *Temporal and Modal Logic*, In Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, Elsevier, 1990.

[27] E. Clarke, "Counterexample-guided abstraction refinement", *Temporal Representation and Reasoning, Fourth International Conference on Temporal Logic*, 2003.

[28] M Nüttgens, T. Feld, V. Zimmermann, "Business Process Modeling with EPC and UML: Transformation or Integration?", *Proceedings of Unified Modeling Language - Technical Aspects and Applications*, 1997.

[29] J. Bacon, K. Moody, W. Yao, “A model of OASIS role-based access control and its support for active security”:, *ACM Trans. Inf. Syst. Secur.*, 2002.

[30] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, “Proposed NIST Standard for Role-Based Access Control”, *ACM Transactions on Information and System Security*, Vol.4, 2001.

[31] DP Guelev, M Ryan, PY Schobbens, *Information Security*, “Model-checking access control policies”, pp219-230, 2004.

[32] M. Drouineaud, A. Luder, K. Sohr, “A role based access control model for agent based control systems”, *Proceedings of the IEEE International Conference on Industrial Informatics*, 2003.

[33] K. Fisler, S. Krishnamurthi, L.A. Meyerovich, M.C. Tschantz, “Change management: verification and change-impact analysis of access-control policies”, *27th Intl. Conf. on Software engineering*, 2005.