A Comparison of Security Analysis Techniques for RBAC Models

Ramadan Abdunabi
Colorado State University
Computer Science Department
rabdunab@cs.colostate.edu

Abstract

Role-Based Access Control (RBAC) is emerging as the de facto authorization model for commercial applications because it simplifies access control management and can be used to represent various types access control policies. Consequently, researchers are extending RBAC to support novel applications. RBAC and its extensions have various features which may conflict with each other. Thus, it is important to analyze the applications to ensure that access control breaches do not occur in an application protected by RBAC policies. Researchers have proposed analysis techniques using formal specification languages including Z, UML supported with OCL, *Alloy*, Coloured Petri Nets (CPN), and Timed Automata (TA) to verify RBAC policies. We briefly examine these analysis techniques and discuss their suitability for RBAC policy verification. One common problem is that most automated approaches do not scale well. Future work in this area should investigate techniques for reducing the state space size and also approaches for reducing the verification times.

Keywords: Formal Specification, Verification, Model Checker, Constraint Rules, RBAC,

1 Introduction

Access control policies protect the resources of an organization by controlling who has access to what objects. Role-based access control (RBAC) model has become the de facto authorization models used by commercial applications because it can model various access control requirements and it simplifies access control management [15, 16]. RBAC and its extensions have various features that may interact with each other resulting in conflicts. Towards this end, there is a need to verify that an application incorporating RBAC policies is indeed adequately protected.

In RBAC [16], *roles*, *users*, *permissions*, and *sessions* are the three primary building components. Roles represent job functions within an organization and permissions for performing actions on protected resources are associated with roles rather than single users. Authorized users are assigned to roles based on their responsibilities and qualifications. Users instantiate a session to activate a subset of roles assigned to them. The user-role assignment and permission-role assignment relations are many-to-many, but the user-session assignment is many-to-one as each session can be associated with only a single user. The lines of authority and functional responsibility structure in an organization is represented in RBAC by role hierarchies *RH*. Role hierarchy is a partial order relation among roles which is reflexive, transitive and anti-symmetric relation. Strictly speaking, Joshi et al. [11] identified two types of role hierarchy relations termed as permission-inheritance hierarchy role-activation hierarchy. Later, researchers recognized the need of supporting constraints on both permissions and roles for modeling and enforcing security policies. A family of security constraints useful for many application domains are defined for RBAC including the principle of Separation of Duties (SoD) and Cardinality constraints. SoD constraints are useful for preventing frauds through conflict of interest violations. SoD constraints ensure that the same user is not assigned conflicting roles and same role is not assigned conflicting permissions. Cardinality constraint imposes restriction on the number of roles assigned to each user, or the number of users assigned to a particular role.

Researchers have also extended RBAC to support novel kinds of applications [2, 6, 11]. For example, in some applications an access to protected resources is contingent upon not only the role of the user but also on contextual information such as current time or location of the user making the access request. Extensions to RBAC include support for temporal constraints [11], spatial constraints [6], and support for spatio-temporal constraints [2].

Along the development of access control models, it is also important to provide a framework of formal verification of security properties of an application using such models. The verification and validation of underlying access control specifications can be performed by manual inspection or by applying formal

analysis techniques. Manual inspection is tedious and error-prone. Towards this end, we investigate the use of formal methods based approach for RBAC verification. Specifically, we investigate approaches based on Z [19], UML with support of OCL [1], *Alloy* [18], Coloured Petri Nets CPN [17], and recently Timed Automata TA [13]. To the best of our knowledge, there is no work that investigates such approaches and exhibits their relevant advantages and shortcomings.

The rest of the paper is organized as follows. Section 2 provides a background of the formal specification languages. Section 3 investigates how the current security analysis techniques have specified and analyzed RBAC models by using the formal specification languages. Section 4 concludes the paper with some pointers to future directions.

2 Background

Z is a formal language [9] suitable for modeling system that can be viewed is state-based transition systems. Using Z, a system can be modeled as a state model along the operations that change such state. It is based on first-order predicate logic. The basic elements of Z are state and operation schema which are a set of deceleration and predicates. A state schema defines an abstract objects, while operation schema defines abstract operations. Declarations are formulated from sets and relations over that sets. In Z notations, a variable represents a set and its type is a set in the power set. A special notation of power set is used to define all possible subsets of a set. Predicates in the state schema define a set of semantic facts/constraints that should not violated under any conditions. In an operation schema, predicates define a set of pre-conditions and post-conditions that should be satisfied before and after operations. $Z/EVES$ [14] is a modeling tool supports Z specifications language for specification and theorem proving in different ways. Domain checking checking feature of $Z/EVES$ ensures the correctness of Z specification syntax and it is meaningful.

UML is an object modeling technique and has a family of models which can be used to specify, visualize, documents objects of a software system. Further, it includes a set of intuitive graphic notation for representing the relationship among various entities in a software system [5]. UML includes a family of isolated models including functional, static and dynamic models. The first model specifies the functional requirements of a software system using use case diagrams. The second is the static model which represents the structural view of data in a system at conceptual, requirements, and implementation level. System entities are replaced by object classes in the UML static diagram. In the static diagram model object classes are defined in terms of names, attributes, behavior, and relationship labelled with multiplicity, constraints, and role names. Such relationship is termed as association among classes and it includes on specialization, generalization, and aggregation relations. Third, the behavior of a system is modeled using dynamic diagrams such as collaboration, sequence, and state-chart diagrams. Collaboration and sequence diagrams show the interaction among object classes to perform a functional requirement. State-chart diagram views the object class states and transitions under different events. OCL is a textual specification language extension to UML to formally specifying constraints on UML object diagrams [12]. It enforces constraints on object values and relationships with conformance to the system requirements. Collections are the main building blocks of OCL which includes on collection types of sets (no duplicates), bags (duplicates), and sequences (duplicates, ordered).

The formal language *Alloy* is a predicate logic and set theoretic inspired by first-order logic. It is suitable to express the structural and behavior of system components [7]. It is capable of expressing various structural constraints and behavior. It provides declaration syntax compatible with graphical object models. *Alloy* declarations syntax is powerful enough to express complex constraints on sets and relations which cause a software to consistently transit from one state to another. Typically, *Alloy* model consists of signature declarations, fields, facts, and predicates, assertions, and scope. Signature declaration is the main paragraph of *Alloy* model where system entities are defined as set of atoms. Atom is the basic entity of the *Alloy* model; it is indivisible, immutable and uninterpreted. In *Alloy* model, the relation among system objects is composed from atoms where each atom is an object. A field defined into a signature to represent a relation between two or more signatures. Two types of relations can be defined in *Alloy*: a singleton relation represent set of scalars, composed relation is a collection of tuples made up of atoms. Facts are invariants defined on signatures and relations that should not be violated in all system states. A Predicate/Function allows users to perform some operations on the system and *Alloy* produce instances that satisfy this predicate (e.g., simulation). *Alloy* assertions check the model for errors and and a counter example of the operation specified in that assertion is generated in a specified scope. Within a particular scope, *Alloy* might not find a counter example unless we increase the scope. A scope defines bounds on each signature so that the instances and counter examples are constructed based on that scope. *Alloy* specification language is supported by a *Alloy* Analyzer software that allows automated analysis of models [8].

CPN language is widely used for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, and non-deterministic [10]. It supports a well-defined semantics that unambiguously specifies the structure and behavior of system components. It represents system compo-

nents in a graphical language. It is a a bipartite graph and mathematical modeling makes it easy to see the basic structure of complex systems (e.g., understand how the individual processes interact with each other. The primary three components: places represent the passive elements of the system, transitions represent the active elements, and arcs interconnect places and transitions. It supports two types of arcs: input arcs connect places with transitions, while output arcs connect transitions with places, and inhibitor arcs which is used to prevent transition. Places can contain tokens that represent the current state of a system. The current marking/state is defined by the distribution of tokens in each place. Transitions are active components model activities which can occur by transition fires, thus changing the state of the system means changing the markings of the CPN. Once transitions are enabled, they can be fired. Firing a transitions means an occurrence of events which changes system state. When a transition is enabled, it can be fired. Transition is enabled only if all constraints associated with event modeled by that transitions and its input arcs are fulfilled.CPN analysis methods uses the occurrence graphs of the $CPNs$ which is also called state spaces or reachability graph that shows all possible reachable marking (system states) starting from the initial system state. Occurrence graph analysis is based on the construction of a weighted directed graph which has a node for each reachable state and an arc for each possible state change (occurring binding elements, i.e different binding between arc expression variable and tokens in each marking). The available CPN supporting tools allow us to automataly construct and analyze occurrence graphs.

Alur and Dill [3] defined the theory of Timed Automata (TA) for modeling and verification of real-time systems. Timed automata is a finite state machine (a finite directed graph containing a set of nodes/locations connected through edges) augmented with non negative real valued variables which model logical clocks in a system. A node in the timed automata represent a control state and edges are annotated with actions, constraints, and variable updates. Clocks are initialized to zero when the system is started, and they advance synchronously at the same pace. Labels on edges represent guards. Clocks are used to express guards on edges that constraint the behavior of the Timed Automata. System transitions are represented by edges that are taken whenever clock values satisfy guards on that edges. Sometimes, the clock values maybe reset to zero whenever a specific transition is taken. Furthermore, control states (locations) are associated with guards termed as invariants that should be satisfied to stay in such states. A network of timed automata is required to model a complete system. Firing transitions in each timed automata may synchronize with one or more transitions in other automata within the network. Timed automata has supporting tool $UPPAAL$ [4] which allow us to perform automated analysis. $UPPAAL$ provides an interactive environment for modeling, simulating, and analyzing of real time systems modeled as timed automata. It allows us to write security queries using Computational Tree Logic (CTL) in an more expressive and intuitive way than other model checkers. Further, $UPPAAL$ offers a variety of optimization techniques reduce the size of state space to improve the model performance.

3 Security Analysis Approaches of RBAC models

We will discuss the proposed five verification approaches [1, 13, 17, 18, 19] of RBAC models based on our proposed classifications. We categorize that security analysis methods on the basis of the characteristics of the forementioned formal specification languages used in such approaches. Some of these specification language are textual based and others are graphical with a support of well-defined syntax and semantics. Further, some of these modeling languages are supported by software tools for performing automated analysis, whereas others are merely modeling languages without tool support.

3.1 Non-Automated Approaches

Among the aforementioned approaches of RBAC models, the security analysis methods that make use of Z specification language and modeling language UML with a support of OCL have limited automated analysis tools. Further, the first verification approach proposed by Chunyang et al. [19] is only mathematical based, while the the second approach by Ahn et al. [1] is graphical based. The following discuss the solutions and problems in both approaches based on these two characteristics.

3.1.1 Non-Graphical Based

In the first study, Chunyang et al. [19] have used Z specification language to specify and verify the consistency of classic RBAC system using theorem prover. The state-based verifiable model represents RBAC components, constraints, and state-transition operations. The software tool kit $Z/EVES$ is used for modeling and theorem prover. The state-based verifiable model of RBAC is given by describing the RBAC state and set of operations over that state. The state schema of RBAC defines its sets and relations along with invariants. A number of operation schemas are defined to specify events. There are three states: *initial state* is the starting state of the system, *secure/consistent state* is a state in which all security requirements

are satisfied, and consistent *state-transition* produced by system transition. The authors showed how to verify the consistency of RBAC under two operations namely $AddNewPerm(r, p)$ which assign permission p to role r and $AddRSSoD(r_1, r_2)$ that add two static conflict roles to the $SSoD$ relation. For example, in the first operation, if the permission p is not in conflict with other permissions assigned to role r, then the permission is assigned and state transition is taken, otherwise the state transition is denied. The security rules proved the consistency of the RBAC state under these operations. The theorem stated that the system remains in secure state if and only if state transition start from secure state and operations comply with state-transition constraints. However, this approach is not completely automated as $Z/EVES$ prover can prove simple theorems. Moreover, the theorem prover requires human-intervention and is difficult to use. The formal semantic of the prover is very difficult to be understandable by users, it is hard to know what action we should take when the prover fails. Other limitation in this study is that only simple and limited number of security constraints (e.g., cardinality and separation of duties) is verified. Verifying the interaction among various features in state-transition RBAC model are not addressed in the study.

3.1.2 Graphical Based

In the security analysis approach proposed by Ahn et al. [1], UML and OCL are utilized to model RBAC and visualize a subset of authorization constraints. The idea is simple, UML is used to visualize the RBAC entities and the relationships over them, and OCL is used to express the authorization constrains on these relationships. The main advantages of the is approach it helps to visualize the potential violation of the RBAC constraints. UML models are also convenient and comprehensible by software developer to specify authorization constraints. Here, authorization constraints specified are categorized: *prohibited constraints*, prevent RBAC components from doing something with respect to security policy (e.g., separation of duties), *obligation/prerequisite constraints*, force an entity in RBAC to do something before being permissible to do another thing, and *cardinality constraints*, is numerical limitation on the authorizations. Note that, the obligation constraints have not been addressed in the aforementioned analysis method [19]. The authors developed a classic UML conceptual static model of RBAC. In the UML model, classes have unique identification names and attributes which depict the basic components of RBAC: User, Role, Permission, and Session. Then the authors show how to specify constraints that regulates the activities of each instance of a class in the model. The study gives examples on how to write OCL expression to specify SoD constraints on user-role assignment(e.g., conflict roles cannot be assigned to the same user) and SoD on permission-role assignment (e.g., conflict permissions are not permissible to be assigned to the same role), SoD on role activations (e.g., conflict roles can be activated in the same session). Despite all the advantages in the approach, some inherited limitations are remaining. The approach is static in nature and dose not capture the dynamic properties of RBAC. Similar to the aforementioned approach, the approach is not supported by automated analysis. Further, the validation of OCL constraints by OCL parser only checks the syntax and type of the expressions. Using UML static diagram model does not guarantee that all conflicts and security violations are detected.

3.2 Automated Approaches

Alloy, CPN, and TA approaches improve upon aforementioned approaches in the sense that they are graphical based approaches with well-defined syntax and semantic. Typically, a graphical representation simplifies complex problems such as specifying and analyzing consistency of RBAC and make it understandable and helps to find inclusive solutions. Further, these later approaches are supported by automated analysis tool that explore the state space of RBAC system for potential existence of inconsistent states.

Toahchoodee and Ray [18] utilized the capability of formal language *Alloy* for modeling and analyzing their spatio-temporal RBAC. It shows how various features in spatio-temporal RBAC can be expressed and analyzed in *Alloy* model. Spatio-temporal model imposes spatio and temporal constraints on user-role assignment, permission-role assignment, user-role activation, role hierarchies, and separation of duties. Here, *Alloy*-based specification of spatio-temporal RBAC is developed. The first part in *Alloy* model describes the primary entities in spatio-temporal model using in deceleration signatures with invariants. Facts/invariants are defined so that they should not be violated under any condition. For example, a user can activate role during the predefined time interval and in location. Constraints are specified in the model using *Alloy predicates*. *Alloy* Analyzer shows an instance in the model that confirms the predicate in particular scope. Finally, the conflict among various features features in model are verified using *Alloy assertions*. The assertions specifies security properties that we want to check. The check command in *Alloy* Analyzer verifies the property in the assertion in predefined scope. If the security property is not satisfied in the mode, *Alloy* Analyzer shows a counterexample which means there exist a state violates this property. Despite all of the strengths in the study, there are some shortcomings. The approach analyze security properties at a model level only. On other words, the security analysis of spatio-temporal RBAC model is independent from

real world applications. Different impact of spatio-temporal RBAC components on the analysis approach is ignored which is important to evaluate the performance. Further, there are some limitations associated with *Alloy* Analyzer. First, it is non-trivial to analyze and understand system behavior using *Alloy*. In particular, it is difficult to identify undesirable states that a system might enter under different situations. Second, *Alloy* is not commonly used for modeling and verifying real time systems. It does not support real variables, thus real valued clocks cannot be modeled. Here, time intervals is represented by a set atoms which are indivisible, immutable and uninterpreted. Third, *Alloy* Analyzer visualize only one counter example even though there might be multiple sources of inconsistencies in the policy. Last but not least, the scope of *Alloy* Analyzer is very sensitive, so it is not suitable to analyze the interaction of many features.

Shafiq et al. [17]. proposed CPN based verification framework to check the correctness of GTRBAC model. It shows how to capture GTRBAC states and events into CPN model thus allowing state-based analysis for security policies. Here, color sets represents the basic elements of GTRBAC with their responding data types. Further, relations such as user-role assignment is defined to be a color product. Places/control states in CPN capture the state GTRBAC model. For example, *EnabledRoles* place in the model only role stores tokens that are in enabled state. Arcs labelled by expressions to model $GTRBAC$ constraints. Once the token marking is changed, it reflects that the state of $GTRBAC$ is changed to secure or inconsistent state. Transitions in the model represents events/operations in $GTRBAC$. CPN representation of user-role assignment and de-assignment event for a role is shown to demonstrate the correctness of the approach. The reachability analysis is performed using Occurrence Graph to identify the existence of security violation. Even though the approach followed by the authors was generally reasonable and convincing, there are gaps that need to be filled. Only cardinality, separation of duty constraints are addressed in the study. It dose not handle the verification of temporal constraints. Further, the approach considers the analysis of security requirements in isolation. The CPN model representation is strictly bounded in order to have finite number of nodes in occurrence graph. When new entities (e.g., tokens) are added, the exhaustive nature of the reachability analysis time increases exponentially. Following this approach, we can not analyze an integrated model of RBAC that combines all features due to the limitation of the reachability analysis. The study does not show how to map $GTRBAC$ components to CPN model using transformation algorithms. One of the limitations in CPN language is that it is static in nature as the initial places need to be initialized with a specific number of tokens which remain stable during the verification process. Security queries are defined by using ML Language which is difficult to formulate and hard to understand by human. For analyzing real time systems timed transitions, timed tokens, and timed arcs need to be defined to model temporal constraints, this increases the state space drastically. Finally, CPN tools do not offer a support of optimization techniques for minimizing search state space and time.

Mandol et al. [13] show how to verify temporal aspects of $GTRBAC$ model [11] using timed automata which subsumed all aspects of their prior approaches. In the approach, $GTRBAC$ system is modeled as a network of timed automata. Primary components of GTRBAC such as roles, users, permissions are mapped to state transition system using timed automata which capture their the state and behavior corresponding entities. Timed automata interact with each other through synchronization channels to represent the behavior. To reduce state space and verification time, only a single global clock is used to specify temporal constraints. A set of desirable security properties are specified as security queries using CTL. Finally, this set of properties are verified against the model using $UPPAAL$ model. Clearly, this analysis approach fills the gaps of the aforementioned approach in terms of the number of temporal constraints and the support of integrated model. In contrast with the previous approach, a set of transformation algorithms are given to describe the construction of timed automata for each component based on policy specifications. For example, a role can be in enabled or disabled state at different time intervals, thus two locations/control are needed to model these states. Locations are connected by directed labelled edges to model system events. The verification is performed for different configuration characterized by number of users, roles, permissions. The results reveal that, with increase of temporal constraints, the versification time and state space size increase drastically. The approach is convincing and improves upon other approaches, but it suffers from a number of limitations. The proposed approach suffers from state space explosion if large number of temporal constraints are considered. The optimization techniques and guidelines for state space reduction supported by $UPPAAL$ have not been used in the approach. Further, user-role activation are classified into four categories namely single role activation, multiple role activation at different time instances, simultaneous role activation, and mutual exclusion role activation. As a result, users cannot be pointed to be in multiple categories in a corresponding system, however, in real world applications, most often users rare authorized to be in multiple categories of role activations at the same time. Finally, the structure of the timed automata need to be changed with change of the entity behavior (e.g., new locations and edges might be added or removed to model a specific behavior of that entity). In contrast, in CPN, we need to only add a new token and the structure of the model remains as it is.

4 Conclusion and Future Work

In this work, we investigated several techniques for analyzing RBAC specifications. Techniques that are easy-to-understand and use are likely to be used by application developers. Moreover, automated approaches not requiring human intervention are likely to perform better. Most of the approaches are limited by the size of the system they can verify.

In future, we plan to investigate scalability issues with respect to access control verification. This includes work on how to abstract the model so that the unimportant details are abstracted away. The other issue we plan to investigate how to partition the problem, analyze each partition, and then argue about the correctness of the entire problem.

References

[1] Gail-Joon Ahn and Michael E. Shin. Role-based authorization constraints specification using object constraint language. In *WETICE*, pages 157–162, 2001.

[2] Subhendu Aich, Shamik Sural, and Arun K. Majumdar. Starbac: Spatio temporal role based access c ontrol. In *OTM Conferences (2)*, pages 1567–1582, 2007.

[3] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[4] Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on uppaal. In *SFM*, pages 200–236, 2004.

[5] Grady Booch, James E. Rumbaugh, and Ivar Jacobson. The unified modeling language user guide. *J. Database Manag.*, 10(4):51–52, 1999.

[6] Maria Luisa Damiani, Elisa Bertino, Barbara Catania, and Paolo Perlasca. Geo-rbac: A spatially aware rbac. *ACM Trans. Inf. Syst. Secur.*, 10(1), 2007.

[7] Daniel Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, 2002.

[8] Daniel Jackson, Ian Schechter, and Ilya Shlyakhter. Alcoa: the alloy constraint analyzer. In *ICSE*, pages 730–733, 2000.

[9] J. Jacky. *The way of Z.* Cambridge University Press, 1996.

[10] Kurt Jensen. An introduction to the practical use of coloured petri nets. In *Petri Nets (2)*, pages 237–292, 1996.

[11] James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[12] Anneke Kleppe and Jos Warmer. An introduction to the object constraint language (ocl). In *TOOLS (33)*, page 456, 2000.

[13] Samrat Mondal, Shamik Sural, and Vijayalakshmi Atluri. Towards formal security analysis of gtrbac using timed automata. In *SACMAT*, pages 33–42, 2009.

[14] Mark Saaltink. The z/eves system. In *ZUM*, pages 72–85, 1997.

[15] Ravi S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *ESORICS*, pages 65–79, 1996.

[16] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[17] Basit Shafiq, Ammar Masood, James Joshi, and Arif Ghafoor. A role-based access control policy verification framework for real-time systems. In *WORDS*, pages 13–20, 2005.

[18] Manachai Toahchoodee and Indrakshi Ray. On the formal analysis of a spatio-temporal role-based access control model. In *DBSec*, pages 17–32, 2008.

[19] C. Yuan, Y. He, J. He, and Z. Zhou. A Verifiable Formal Specification for RBAC Model with Constraints of Separation of Duty. In *Information Security and Cryptology*, pages 196–210. Springer, 2006.