Security Analysis of Temporal-RBAC using Timed Automata

Samrat Mondal and Shamik Sural
School of Information Technology
Indian Institute of Technology, Kharagpur, WB, Pin- 721302
email: {samratm, shamik}@sit.iitkgp.ernet.in

Abstract

Role Based Access Control (RBAC) is arguably the most common access control mechanism today due to its applicability at various levels of authorization in a system. Time-varying nature of access control in RBAC administered systems is often implemented through Temporal-RBAC - an extension of RBAC in the temporal domain. In this paper, we propose an initial approach towards verification of security properties of a Temporal-RBAC system. Each role is mapped to a timed automaton. A controller automaton is used to activate and deactivate various roles. Security properties are specified using Computation Tree Logic (CTL) and are verified with the help of a model checking tool named Uppaal. We have specifically considered reachability, safety and liveness properties to show the usefulness of our approach.

Keywords: Temporal-RBAC, Timed Automata, Security Analysis, Model Checking, CTL

1. Introduction

In an RBAC system [1], roles are created to perform certain job functions. Users are assigned to various roles depending upon their responsibilities and qualifications. An active role is associated with one or more permissions. Recently, several attempts have been made to develop variants of basic RBAC to handle temporal, spatial and spatio-temporal contexts [2], [3], [4]. TRBAC model [2] allows temporal constraints on role activation and deactivation. It also permits periodic role enabling and disabling and allows temporal dependencies among such actions.

Along with the development of access control models, it has also become necessary to perform security analysis of various models [5] [6] [7]. Very recently, Jha et al. [8] have proposed a method for formal verification of RBAC policies. However, to the best of our knowledge, there is no reported work on specification and formal verification of security properties in Temporal-RBAC administered systems. In this paper we propose a methodology for security analysis of Temporal-RBAC systems. Each role is mapped to a timed automaton, which is instantiated for every user taking up that role. Model checking is used for formal verification of security properties expressed in Computation Tree Logic (CTL).

2. Related Work

The safety analysis problem was first formulated in access control matrix model by Harrison et al. [5]. They have shown that the problem is *undecidable*. Koch et al. in [9] proved that it is possible, with reasonable restrictions on the form of the rules, to obtain access control models where safety is decidable. Of late, Li et al. [7] have studied security analysis problem in AATU (Assignment And Trusted Users) and AAR (Assign And Revocation). In [8], Jha et al. have compared the use of model checking and first order logic programming for the security analysis of RBAC system and concluded that model checking is a quite promising approach for security analysis.

Bertino et al. [2] have provided a polynomial algorithm to verify whether specifications in TRBAC are free from ambiguities or not. But not much work has been done on the security analysis of a Temporal-RBAC system. The model of a Temporal-RBAC system needs to express the continuous change of time and related temporal constraints in a formal and convenient way. To design such models, we propose to use timed automata (TA). A timed automaton can describe and analyze the behavior of a finite state system with real valued clocks. It allows time transitions i.e., state change can occur if certain temporal constraints are satisfied [10]. Each state of such a system is represented by a location and a non-negative real valued clock pair. For verification of properties, the state-space of the system is to be stored. The state-space of a timed automaton is clearly infinite since clocks are variables that range over the set of real numbers. This is known as *state space explosion problem*. It has been shown that most verification problems on timed model are *PSPACE-hard*. But an

978-0-7695-3324-7/08 $25.00 © 2008 IEEE
DOI 10.1109/IAS.2008.10

important observation [11] shows that reachability problem becomes *NLOGSPACE-complete* with one clock TA. In our proposed methodology we utilize this result and build the Temporal-RBAC system using a single clock.

In this work, we enforce certain restrictions on our model. For example, temporal constraints can be applied only on role activation and deactivation. We also assume that one global clock is used in the system. While it is well understood that a complete Temporal-RBAC implementation may not be so restrictive, this being the first attempt in Temporal-RBAC property verification, we have tried to establish a potential methodology for performing security analysis.

3. Security Analysis of Temporal-RBAC

For the security analysis of Temporal-RBAC, we propose timed automata based model checking mechanism. The security properties are expressed in terms of Computational Tree Logic (CTL). In the next few subsections, we give the detailed descriptions of our approach.

3.1. Brief Overview of Timed Automata

A timed automaton is a finite state machine equipped with a set of clocks. All clocks are synchronized, i.e., they all advance at the same pace. Transitions are associated with a guard which is a predicate over the clocks. The guard determines when a transition can take place. A timed automaton can be formally defined as a 6-tuple (L, l_0, C, A, E, I) where L is a set of locations or control states, l_0 is the initial state, C is the set of clocks, A is the set of actions, E is the edge between two locations and can be defined as $E = L \times A \times B(C) \times 2^n \times L$. Here $B(C)$ is a set of clock constraints or guards and 2^n is a set of clocks to be reset where n is $|C|$. I assigns invariants to locations and can be written as $I : L \rightarrow B(C)$.

3.2. Formal Representation of Temporal-RBAC using Timed Automata

The overall flow of our approach consists of the following steps. After analyzing a given Temporal-RBAC system, we construct a model using timed automata (TA). The desirable properties are specified using CTL. Model checking method is then applied to verify the properties. If all the properties are satisfied, then the system can be considered as "safe". Otherwise, the system is considered to be "unsafe". Three basic components of an RBAC system are - role, permission and user represented by the three sets R, P and U, respectively. Role is associated with user by the role-user assignment (UA) relation and with permission through role-permission assignment (PA) relation [1]. For Temporal-RBAC, temporal constraints are also to be addressed in the model. In our analysis approach, we represent each role by a parameterized timed automaton.

3.2.1 Construction of TA

We have developed two algorithms to map a Temporal-RBAC system to TA based system. Algorithm 1 describes how to construct a timed automaton corresponding to each r where $r \in R$ i.e., the set of roles. Algorithm 2 represents the construction of Controller TA which is an automaton that controls which role is to be activated or deactivated depending on different time constraints.

Algorithm 1 Constructing TA from R

1: **for** each member $r_j \in R$ where $j = 1, \cdots, |R|$ **do**
2: construct a timed automaton T_j with a parameter u where u represents users assigned to the role r_j.
3: T_j should have at least two locations named $Idle\_R_j$ and $Active\_R_j$.
4: The edge from $Idle\_R_j$ to $Active\_R_j$ has a synchronization label $activate\_R_j?$ and the edge from $Active\_R_j$ to $Idle\_R_j$ has a synchronization label $deactivate\_R_j?$
5: **end for**

Algorithm 2 Constructing Controller TA

1: Construct a timed automaton say T_c with initial location L_0.
2: sort the role activation-deactivation time constraints of the form $x \bowtie c$.
3: let the sorted time constraints are in set A and let the length of A be n.
4: let $S = L_0$
5: **for** $i = 1$ to $(n - 1)$ **do**
6: add a location L_i in T_c such that the edge from S to L_i is labeled with the corresponding time constraint and labeled with $activate\_R_i!$ if it is an activating time constraint or by $deactivate\_R_i!$ if it is a deactivating time constraint.
7: $S = L_i$
8: **end for**
9: add the edge from S to L_0 with the last time constraint and corresponding activation or deactivation synchronization label. The clock is to be reset with this time transition.

In our approach, the total number of processes i.e., number of instantiations of timed automata, N will be $N = |UA| + 1$ i.e., one process is required for each member of UA and one more is required for Controller TA.

3.2.2 Temporal Properties for Verification

In this section, we specify the Temporal-RBAC properties using appropriate logic so that they can be used for verification. First order predicate logic is not capable of expressing the properties with keywords "eventually", "always", etc., which are frequently used constructs in a time varying system. Temporal logic is the best option as it has system of rules and symbolism in terms of time. The operators of temporal logic can be conveniently used to represent statements such as "A role will always activate another role", "A role will be deactivated eventually", etc. It has two variants - *Linear Temporal Logic* (LTL) and *Computation Tree Logic* (CTL). We apply CTL in which two common operators - $\Diamond$ and $[]$ are used to describe the state formulae. $\Diamond$ indicates *at some states* and $[]$ indicates *at all states*. Path formulae quantify over the path or trace and can be used to specify the reachability, safety and liveness properties. Two common operators used in the path formula are *A*, which means *for all paths*, and *E*, which means *there exist some paths*.

Generally **reachability properties** are represented by the expressions of the form $E\Diamond\phi$, which ask whether there exists a path starting at the initial state, such that ϕ is eventually satisfied along that path. **Safety properties** specify that "something bad never happens" or "something good will invariantly be true in all states". So if ϕ be a state formula and if we want to express that ϕ will be true in all reachable states, then we express it as $A[]\phi$. If we want to ask whether there exists a path in which all the states satisfy ϕ, then we use the expression $E[]\phi$. Safety properties are complemented by liveness properties. **Liveness properties** assert that "something good will eventually happen". For example, a liveness property can be that a role will eventually be active. So if ϕ is a state formula, then $A\Diamond\phi$ says that for all paths, ϕ is eventually satisfied at some states. In another form of liveness property, if ϕ and ψ are two state formulae, then $\phi \rightarrow \psi$ represents that if ϕ is true then eventually ψ is also true.

4. Verification and Analysis

We use a tool named Uppaal [12] to design the TA of an Temporal-RBAC system and verify the properties specified in temporal logic. As an example, consider a Temporal-RBAC system in which there are two roles - FullTime Employee and PartTime Employee. We consider a global clock T. FullTime employees are activated when $T = 10$ time unit and PartTime employees are activated when $T = 12$. The PartTime and FullTime employees are deactivated when $T = 16$ and $T = 17$ respectively.

In Figure 1, we show the Controller timed automaton that controls the activation and deactivation of these two roles. When some temporal constraints are satisfied, then the controller automaton sends synchronization signals. These signals are received by the TA representing different roles. The TA for FullTime and PartTime roles can be seen in Figures 2 and 3 respectively. These TA are constructed following Algorithms 1 and 2. For each member of UA relation, an automaton is instantiated. During the verification process these automata act in parallel. Each automaton has two locations, one representing *Idle* and the other representing *Active*. The synchronization label $activate\_ft?$ indicates that the transition from *Idle* to *Active* takes place after receiving the $activate\_ft$ signal from the Controller timed automaton. In this example all the users of a particular role are activated and deactivated at a specific time. Whenever $activate\_ft!$ is fired, all the edges having synchronization label $activate\_ft?$ are synchronized with the emitting process. Here, all the users of FullTime role are activated at T=10.

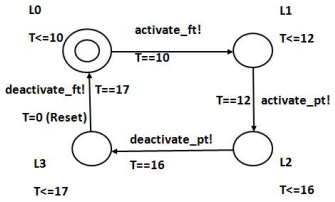

Figure 1. Controller timed automaton

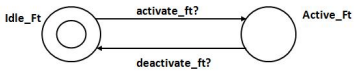

Figure 2. Timed automaton for FullTime role

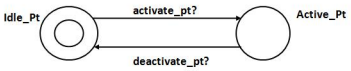

Figure 3. Timed automaton for PartTime role

Our next goal is to specify some temporal properties which will be verified using the model. We consider the following CTL properties.

$$Q_1 : A[]forall(i : id\_pte)PT\_Emp(i).Active\_PT$$

$$imply\ forall(j : id\_fte)FT\_Emp(j).Active\_FT$$

Query Q_1 is a safety property which specifies that in all paths and at all states - if PartTime employees are active then FullTime employees are already active.

$$Q_2 : FT\_Emp(0).Active\_FT \rightarrow FT\_Emp(0).Idle\_FT$$

i.e. an active FullTime employee with id=0 eventually becomes idle also. This is a liveness property.

$$Q_3 : E <> forall(i : id\_fte)\ FT\_Emp(i).Active\_FT$$

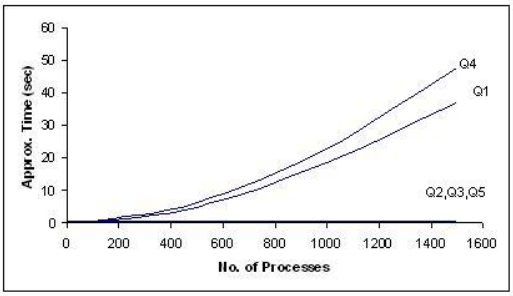

Figure 4. Verification time vs. no. of TA

i.e., there exists a path in which at some states all FullTime employees will be active. This is a reachability property.

$$Q_4 : A[]forall(i : id\_pte)\ PT\_Emp(i).Active\_PT\ \&\&$$

$$forall(j : id\_fte)FT\_Emp(j).Active\_FTimply\ i\ !=\ j$$

This is again a safety property. It ensures that in all paths and in all states, all user with a specific id cannot be a member of both FullTime and PartTime roles.

Finally to check that the system is deadlock free, we use the following tool specific query.

$$Q_5 : A[]\ not\ deadlock$$

To carry out verification of the above five properties we use the following options of Uppaal - breadth first search order, conservative state space reduction, Difference Bound Matrix (DBM) state space representation, automatic extrapolation, no diagnostic test and 16 MB hash table size. The time required for verification of the queries on an Intel 1.5 GHz processor is illustrated in Figure 4. In this figure, number of processes denotes total number of instantiations of all TA. It is seen that the queries Q_2, Q_3 and Q_5 can be verified in almost constant time irrespective of the number of processes. The actual values are quite small and almost merge with the X - axis. For Q_1 and Q_4, verification time shows quadratic variation.

5. Conclusions and Future Work

We have discussed how to represent a Temporal-RBAC system using TA. Two algorithms have been proposed - one to construct a timed automaton for each role and another to construct a controller timed automaton. We have focused on the temporal constraints of user-role assignment relation only and used a single clock to represent the system. While this limits the expressive power of the system, we achieve polynomial time verification result. In general, the verification problem of a time based system is PSPACE-hard. However, use of single clock ensures that we can verify if a system satisfies the desirable properties in quadratic time. In future, we will handle temporal constraints on role-permission assignment relation and incorporate more number of clocks into the system and then verify the properties.

References

[1] R. Sandhu, E. Coyne, H. Feinstein and C. Youman. Role-based access control models. *IEEE Computer*, Vol. 29, No. 2, pp. 38−47, Feb 1996.

[2] E. Bertino and P.A. Bonatti. TRBAC: A temporal role based access control model. *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pp. 191−223, Aug 2001.

[3] E. Bertino, B. Catania, M. L. Damiani and P. Perlasca. GEO-RBAC: A spatially aware RBAC. *10th ACM Symposium on Access Control Models and Technologies*, pp. 29−37, 2005.

[4] S. Aich, S. Sural, A. K. Majumdar. STARBAC: Spatio temporal role based access control. *2nd International Symposium on Information Security*, pp. 1567-1582, 2007.

[5] M. A. Harrison, W. L. Ruzzo and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, Vol. 19, No. 8, pp. 461−471, Aug 1976.

[6] A. K. Jones, R. J. Lipton and L. Snyder. A linear time algorithm for deciding security. *17th Annual IEEE Symposium on Foundations of Computer Science*, pp. 33−41, Oct 1976.

[7] N. Li and M.V. Tripunitara. Security analysis in role based access control. *ACM Transactions on Information System Security*, Vol. 9, No. 4, pp. 391−420, Nov 2006.

[8] S. Jha, N. Li, M. Tripunitara, Q. Wang, W. Winsborough. Towards formal verification of role based access control policies. *IEEE Transactions on Dependable and Secure Computing*, Vol. 5, No. 2, Apr 2008 (to appear).

[9] M. Koch, L. V. Mancini, F. Parisi-Presicce. Decidability of safety in graph-based models for access control. *7th European Symposium on Research in Computer Security*, pp. 229−243, 2002 .

[10] J. Bengtsson and W. Yi. Timed Automata: semantics, algorithms and tools. *Lecture notes on Concurrency and Petri Nets*, LNCS, Springer-Verlag, pp. 87−124, Jul 2004.

[11] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. *15th International Conference on Concurrency Theory*, LNCS, Springer-Verlag, pp. 387−401, 2004.

[12] G. Behrmann, A. David and K.G. Larsen. A tutorial on Uppaal. *4th International School on Formal Methods for the Design of Computer, Communication and Software Systems*, LNCS, Springer-Verlag, pp. 200−236, 2004.