Verification of Spatio-Temporal Role Based Access Control using Timed Automata

Emsaieb Geepalla
School of Computer Science
University of Birmingham
Birmingham, UK
Email:E.M.E.Geepalla@cs.bham.ac.uk

Behzad Bordbar
School of Computer Science
University of Birmingham
Birmingham, UK
Email:B.Bordbar@cs.bham.ac.uk

Kozo Okano
Information Science and Technology
Osaka University
Suita, Osaka Japan
Email:Okano@ist.osaka-u.ac.jp

***Abstract*—The verification of Spatio-Temporal Role Based Access Control policies (STRBAC) during the early development life cycle improves the security of the software. It helps to identify inconsistencies in the Access Control policies before proceeding to other phases where the cost of fixing defects is augmented. This paper proposes a formal method for an automatic analysis of STRBAC policies. It ensures that the policies are consistent and conflict-free. The method proposed in this paper makes the use of Timed Automata to verify the STRBAC policies. This is done by translating the STRBAC model into Timed Automata, and then the produced Timed Automata is implemented and verified using the model checker UPPAAL. This paper presents a security query expressed using TCTL to detect inconsistency caused due to the interaction between STRBAC policies. With the help of an example, this paper shows how we convert STRBAC model to Timed Automata models and verify the resulting models using the UPPAAL to identify an erroneous design.**

I. Introduction

The recent advances in mobile computing, wireless networks and other technologies involved in remote accessing of resources has prompted an urgent need for the creation of Access Control system which takes into consideration the location of the user and the time of access. Such information is essential for controlling various spatio-temporal sensitive applications, which rely on the Access Control mechanism, in organizations. For example, the access to some resources at an organisation could be permissible only at a specific time and a specific location. In order to be adaptable to the requirements of such applications and technologies with both spatial constraint and temporal constraint, several Access Control models have been proposed [3], [7], [8], [9]. The STRBAC model is one of the Access Control models that has been presented to cater for the need of context information [9]. The STRBAC model incorporates various polices, such as, Role Hierarchy, Separation of Duties constraints, and Cardinality constraints.

It is possible that STRBAC polices conflict with each other, in particular when various policies interact with each others. A typical situation is when User Role Assignment and Role Hierarchy cause a violation of Separation of Duty constraints. For example, an organisation may require that the same user should not be assigned to the two conflicted roles `Teller` and `Loan Officer`, at the same time and same location, whereas it requires that a user u be the `System Operator Manager` and the role `System Operator Manager` is a senior role to the `Teller` and `Loan Officer` at any time and any location. Such situations are often referred to as inconsistencies in Access Control policies. This example is inconsistent because the user u can assign to the two conflicted roles; `Teller` and `Loan Officer`, at the same time and same location because he/she is assigned to the role `System Operator Manager` which is a senior to the roles `Teller` and `Loan Officer`. Such scenario could pose dangerous security issues that could even cause the downfall of the organization [2]. It is therefore essential to perform an analysis of STRBAC policies to identify inconsistencies in the polices.

In this paper we propose a formal method to perform an automated analysis on STRBAC policies in order to detect inconsistencies. To achieve this, we shall first model the STRBAC using Timed Automata. To do so, we shall translate some of the STRBAC features such as Users, Roles, Times, Location and User Role Assignment to Timed Automata models and the other features of STRBAC will be expressed using Timed Computation Tree Logic (TCTL). Then we shall implement and verify the Timed Automata models and the TCTL using UPPAAL which provides an interactive environment for modelling, simulating, and analysing of real time systems modelled as Timed Automata.

The remainder of this paper is organized as follows. Section 2 provides a review of STRBAC model, Timed Automata as well as a brief introduction to UPPAAL. In section 3 we introduce an example, which will be used to describe our approach. Section 4 describes our effort to use Timed Automata to analyse STRBAC policies to detect inconsistencies. Section 5 briefly presents related works that have motivated this research. The paper ends with a conclusion and future work.

II. Preliminaries

This section provides a brief introduction to Spatio Temporal Role Based Access Control model (STRBAC), Timed Automata as well as a brief introduction to UPPAAL.

A. Spatio Temporal Role Based Access Control

Several Spatio-Temporal Access Control model have been presented recently to cater for the needs on many mobile application [2], [7], [8], [9]. Our metaphor of Spatial Temporal Access Control is based upon recent work of Inderakshi et al. [10]. There, Access Control is governed by the time and the location conditions, in which the right of assigning a user to a role and permissions owned by that role is banked on spatial

978-1-4673-4723-5/12/$31.00 ©2012 IEEE

and temporal information. In this paper we restrict ourselves to STRBAC without sessions and delegation. If there is no chance of confusion we might use the words Access Control instead of STRBAC in the rest of the paper.

1) The Basic Concepts of the STRBAC: The basic concept of the STRBAC model shown in Fig.1, consists of the following five component sets: Users (U), Roles (R), Permissions (P), Times (T) and Locations (L) and the following two relations sets: User Role Assignment (URA) and Permission Role Assignment (PRA).

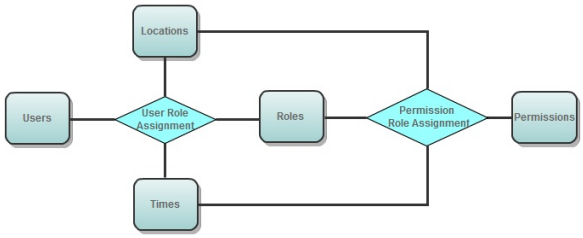

Fig. 1. Basic Concepts of STRBAC Model

- U, R, P, T, L are respectively finite sets of users, roles, permissions, times and locations
- **User Role Assignment**: URA is a relation that associates users with roles based on the time and location, $\text{URA} \subseteq \text{U} \times \text{R} \times \text{T} \times \text{L}$. This means users can be assigned to a set of roles at different points of time and location and every role might be assigned to one user or more users at different points of time and location. We write $\text{URA}(u, r, t, l)$, meaning that a user u is assigned to a role r at time t and location l.
- **Permission Role Assignment**: PRA is a relation that associates roles with permissions based on the time and location, $\text{PRA} \subseteq \text{R} \times \text{P} \times \text{T} \times \text{L}$. This means roles can be assigned to a set of permissions at different points of time and location and every permission might be assigned to one role or more roles at different points of time and location. We write $\text{PRA}(r, p, t, l)$, meaning that a role r is assigned to a permission p at time t and location l.

2) Role Hierarchy (RH) in STRBAC: RH is a partial order on the set of roles, $\text{RH} \subseteq \text{R} \times \text{R} \times \text{T} \times \text{L}$. We write $r_i \succeq r_j$ meaning that the role r_i is a senior to the role r_j at any time and any location. This means r_i inherits all the permissions of r_j, and if there is a user assigned to senior role r_i then he/she could also assign to the junior role r_j. RH could be unrestricted, time dependent, location dependent, or time and location dependent and written as $\succeq$, $\succeq_t$, $\succeq_l$ and $\succeq_{t,l}$ respectively.

3) Separation of Duty between Role (SoDR) in STRBAC: SoDR is a constraint over roles, which specify that users should not assign to exclusive roles, $\text{SoDR} \subseteq \text{R} \times \text{R} \times \text{T} \times \text{L}$, We write $\text{sodr}(r_i, r_j, t, l)$ meaning that the two exclusive roles r_i and r_j should not be assigned by the same user at time t and location l. SoDR can be unrestricted $\text{sodr}(r_i, r_j)$, time dependent $\text{sodr}(r_i, r_j, t)$, location dependent $\text{sodr}(r_i, r_j, l)$, or time and location dependent $\text{sodr}(r_i, r_j, t, l)$.

B. Timed Automata

Alur and Dill [13] introduced the idea of Timed Automaton. A Timed Automaton is finite-state machine extended with clock variables. It uses a dense-time model where a clock variable evaluates to a real number. All the clocks progress synchronously. A system is modelled as a network of several such Timed Automata in parallel. The state of a system is defined by the locations of all automata, the clock constraints, and the values of the discrete variables. Every automaton may fire an edge (sometimes called a transition) separately or synchronize with another automaton, leading to a new state. Timed Automata are extensions of the conventional automata with variable and constraints for expressing real-time dynamics. They are widely used in the modelling and analysis of real-timed systems. For more information on TA the reader is referred to [8], [13].

C. UPPAAL

UPPAAL is a famous model checker for extended Timed Automata by Yi-Wang et al. [1], [6]. It also supports model checking for the conventional Timed Automata. UPPAAL allows verification of expressions described in an extended version of CTL. In addition, it supports local and global integers and primitive operations on integers, such as addition, subtract and multiplication with constants. Such expressions are also allowed on the guards of transitions. The model of the system can be created from multiple Timed Automata which are synchronised together via a CCS-like synchronisation mechanisms [6]. For more information on TA the reader is referred to [1], [6].

III. Running Example: SECURE Bank System

In this section we introduce a simple Banking application taken from [4] to illustrate our approach. The security policies for the SECURE Banking system as given below.

A. Security Policies

1) User Role Assignment: Users are assigned to roles in the Banking system as illustrated in Table 1.

2) Permission Role Assignment: Roles are assigned to permissions in the Banking system as illustrated in Table 2.

3) Role Hierarchy: Some roles in the Banking system are related using unrestricted Role Hierarchy as follows: Role Hierarchy= {(SOM, Teller, DayTime, office1), (SOM, Loan Officer, DayTime, office1), (SOM, DSO, DayTime, office2), (SOM, NSO, NightTime, office2)}.

4) Separation of Duty between Roles: There is only one separation of duty constraint in the SECURE Bank system. Separation of Duty Over Roles={(Teller, Loan Officer, DayTime, office1)}

IV. Modelling STRBAC Model using Timed Automata

The aim of this section is to transform STRBAC model into Timed Automata models. Our work is motivated by the idea that an Access Control system designer should be able to automatically analyse the system prior to its deployment.

TABLE I. USER ROLE ASSIGNMENT CONSTRAINTS

Users	Roles	Times	Locations
Dave	Teller	DayTime (9:00-16:00)	office1
Sarah	Loan Officer	DayTime (9:00-16:00)	office1
John	DayTime System Operator (DSO)	DayTime (9:00-16:00)	office2
John	NightTime System Operator (NSO)	NightTime (16:00-9:00)	office2
Mark	System Operator Manager (SOM)	DayTime (9:00-16:00)	office1/office2

TABLE II. PERMISSION ROLE ASSIGNMENT

Roles	Permissions	Times	Locations
Teller	WRTF: Read and Write Teller Files	DayTime (9:00-16:00)	office1
Loan Officer	WRLF: Read and Write Loan Files	DayTime (9:00-16:00)	office1
DSO	WRSOF: Read and Write System Operator Files	DayTime (9:00-16:00)	office2
NSO	WRSOF: Read and Write System Operator Files	NightTime (16:00-9:00)	office2

We achieve this by mechanically translating STRBAC model into Timed Automata models and then by verifying these models using UPPAAL. The main challenge in this work is finding efficient ways of translating Access Control policies into compact, manageable models. This is done through a number of transformations showing in Fig 2, and described below with the help of the example presented in section 3.

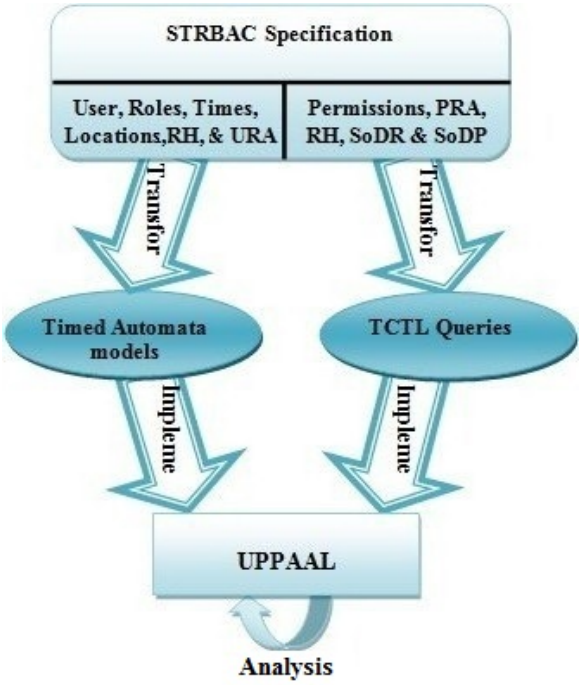

Fig. 2. Our Approach

A. Modelling the Bank System using Timed Automata

The first step of the analysis of Access control policies (i.e. the Bank System) using Timed Automata is to transform the policies into Timed Automata. Some features of STRBAC will be mapped to Timed Automatons. The set of Times which is one of the basic components of STRBAC model is mapped to Timed automaton called Time. For example, in the SECURE Bank system the Time is divided into two periods `DayTime` that is from 9:00 to 16:00, and `NightTime` from 16:00 to 9:00. This will be mapped to a Timed Automaton as depicted in Fig 3.

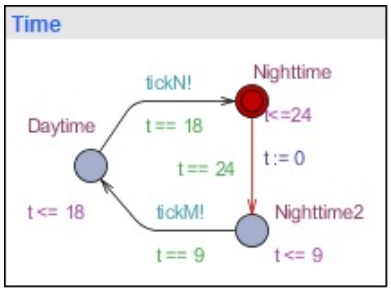

Fig. 3. Timed Automaton for Times

The set of Users which is another component of the STRBAC will be mapped to Timed Automatons. Each Timed Automaton represents a user and the User Role Assignment information for that user such as the roles that he/she could has and the spatial and temporal constraints. For example, in the SECURE Bank system the user `Dave` and the User Role Assignment information for `Dave (Dave. teller, DayTime, office1)` will be mapped to a Timed Automaton as illustrated in Fig 4. The figure shows that the assignment between the role `Teller` and the user `Dave` is true only when the time is `DayTime` and the location is `office1`.

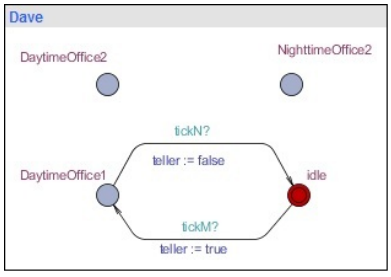

Fig. 4. Timed Automaton for Dave

Similarly the other users `John`, `Sarah` and `Mark` and the User Role Assignment information for those users will be mapped to Timed Automatons as illustrated in Fig 5, 6 and 7 respectively.

Some features of the STRBAC model such as Permission Role Acquire will be expressed using TCTL, and then they will be used as sub-expressions of queries for Timed Automata. Firstly, we shall show how the set of Permission Role Acquire presented in Table 2 is represented using TCTL. The first

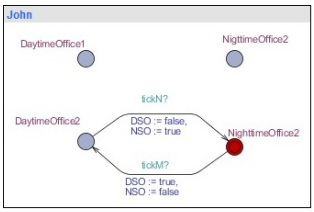

Fig. 5. Timed Automaton for John

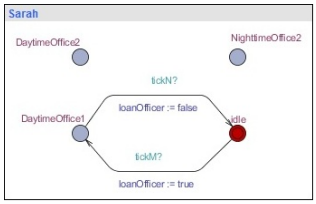

Fig. 6. Timed Automaton for Sarah

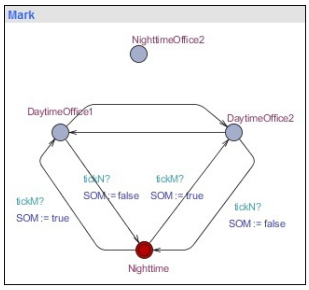

Fig. 7. Timed Automaton for Mark

element of Permissions Role Acquire set `(Teller, RWTF, DayTime, office1)` will be expressed using TCTL as follows:

```
RWTF(u):=u.DayTime_office1 & u.Teller
RWTF:=$\bigvee_{u \in Users}$ RWTF$(u)$
```

This means any user `u` who is a Teller can have the permission `Read and Write Teller File`, where Users is a finite set, so that we can have a finite expression of `RWTF`. For example in the Bank system we have a finite set of Users consists of the following four users: `Dave`, `Sarah`, `John` and `Mark`. This will result in a finite set of expression of `RWTF` as follows:

```
RWTF:= Dave.daytime_office1 & Dave.Teller
  || Sarah.DayTime_office1 & Sarah.Teller
  || John.DayTime_office1 & John.Teller
  || Mark.DayTime_office1 & Mark.Teller
```

Similarly the Permission Role Acquire `(Loan Officer, RWLF, DayTime, office1)` will be expressed using TCTL as follows:

```
RWLF(u):=
    u.DayTime_office1 & u.LoanOfficer
RWLF:=
    $\bigvee_{u \in Users}$ RWLF$(u)$
```

The Permission Role Acquire `(DSO, RWSOF, DayTime, office2)` will be expressed using TCTL as follows:

```
RWSOF1(u):=
     u.DayTime_office2 & u.DSO
RWSOF1:=
     $\bigvee_{u \in Users}$ RWSOF1$(u)$
```

The permission Role Acquire `(NSO, RWSOF , NightTime, office2)` will be expressed using TCTL as follows:

```
RWSOF2(u):=
     u.NightTime_office2 & u.NSO
RWSOF2:=
     $\bigvee_{u \in Users}$ RWSOF2$(u)$
```

The Permission Role Acquire will be a conjunction of `RWTF`, `RWLF`, `RWSOF1` and `RWSOF2`.

The Role Hierarchy can be expressed using Timed Automata and TCTL. As described in section 2 the Role Hierarchy between roles means a user `u` who has the senior role can inherit the junior role and all the permissions assigned to the junior role and the senior role can inherit all the permissions assigned to the junior role. This means new User Role Assignment and Permission Role Acquire will be added to the specification due to the effect of Role Hierarchy. For example the following Role Hierarchy `{(SOM, Teller, DayTime, office1)` means that if there is a user `u` who is assigned to the senior role `SOM` during the `DayTime` and at `office1`, then that user can inherit the junior role `Teller` and all the permission assigned to it such as the permission `RWTF`. In addition to that, the senior role `SOM` can inherit all the permissions assigned to the junior role `Teller`.The new user to role assignment information `(Mark, Teller, DayTime, office1)` will be expressed using Timed Automaton. More precisely, the Timed Automaton for the user `Mark` will be updated with new assignment information as illustrated in Fig 8. While the new permission to role assignment information `(SOM, RWTF, DayTime, office1)` will be represented using TCTL as follows.

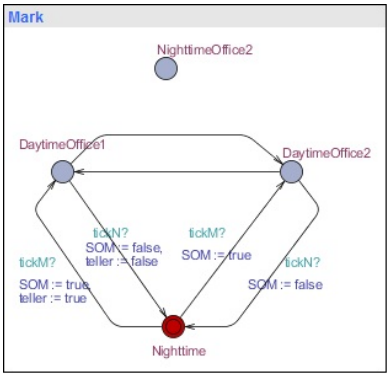

Fig. 8. New Updated Timed Automaton for Mark

```
RWTF2(u):=u.DayTime_office1 & u.SOM
RWTF2:=$\bigvee_{u \in Users}$ RWTF2$(u)$
```

The Separation of Duty between Roles can also be expressed using TCTL. For example the the following Separation of Duty between the Roles `(Teller, Loan Officer, DayTime, office1)` will be expressed using TCTL as follows:

```
SoDP:=And_{u \in Users}(u.DayTime_office1
    implies NOT (u.Teller & u.LoanOfficer)
```

This means there should be no user in the set of Users that is assigned to the two conflicted roles `Teller` and `Loan Officer` during `DayTime` and at `office1`.

B. Model Verification

In the previous section, we have shown the translation between the STRBAC model and the corresponding Timed Automata. However, this work would be incomplete without demonstrating how we can verify the produced Timed Automata models to detect inconsistencies. In This section, we provide a brief description of the automatic verification task by introducing a security query that will be implemented using UPPAAL to be used to verify the Timed Automata models.

Before starting the automatic verification to detect inconsistencies, we must establish what is meant by inconsistencies in STRBAC policies. The STRBAC policies is called inconsistent when, for specific situations different incompatible policies can apply i.e. if User Role Assignment rule is conflicted with the Separation of Duty between Roles. To ensure the consistency of the SECURE Bank system we have formalised several security query using TCTL. The following statement is an example of the TCTL queries that we have formalised.

```
A[] ((RWTF & RWLF & RWSOF1 & RWSOF2 &
      RWTF2 & RWLF2 & RWSOF3 & RWSOF4)
      implies  SoDR)
```

The above security query means that if all the Permission Role Acquire and the Role Hierarchy statements such as `RWTF`, `RWLF`, `RWSOF1`, `RWSOF2`, `RWTF2`, `RWLF2`, `RWSOF3` and `RWSOF4` hold, then `SoDR` statement should also holds, for any time and location configuration. To check whether all of these statements hold or not we have implemented the Timed Automata network presented in section 4.1 and the above security query using UPPAAL. The execution of the verifier of UPPAAL has found that the property is not satisfied as depicted in Fig 8. This means the SECURE Bank system is inconsistent. This is because the statements `RWTF2` and `RWLF2` conflict the `SoDP`. This is because the the model checker UPPAAL has found that the two conflicted roles `Teller` and `Loan Officer` has been assigned by the user `Mark` during the `DayTime` and at `office1`, which is not permissible by the `SoDR`.

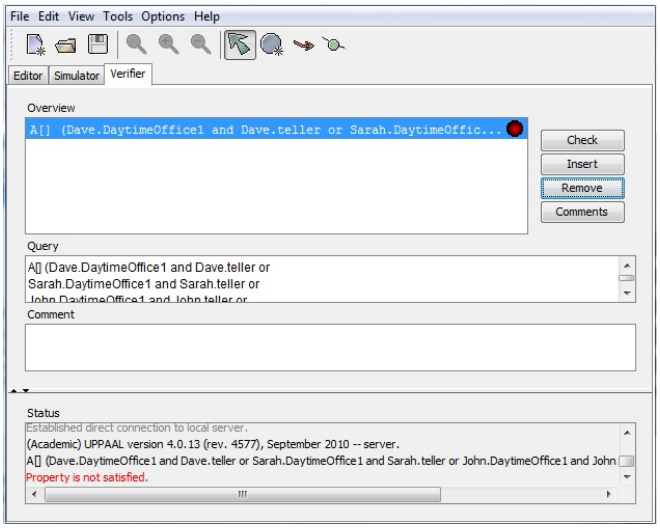

Fig. 9. Verification of the SECURE Bank System via UPPAAL

C. Challenges of Using Timed Automata

Although, our approach makes it possible to perform an automated analysis of the STRBAC model, there is a major limitation of using this approach for verification of a large scale STRBAC model. This is because we have found that the security query that we have built to check one `SoDP` statement is very large queries. This is because the security query has to be verified against the whole STRBAC specification such as Permission Role Acquire and Role Hierarchy. Therefore, verifying a large scale STRBAC specification using this approach will be challenging, specially if the transformation between STRBAC and Timed Automata is carried out manually.

V. Related Work

This section briefly outlines the relevant work done by other people in the field of Analysis of Access Control policies. The use of formal methods in Analysis of Access Control has introduced for a while and there are several other modelling languages, besides Timed Automata [12], that have been used to model and analyse Access Control policies such as Alloy [3], [4], [10] and and Petri Net [11].

The use of Timed Automata in vitrification of Access Control policies is not novel [12]. In [12], Samrat et al. demonstrate how to analyse the properties of GTRBAC model using Timed Automata. The authors show how to transform GTRBAC into Timed Automata models, and then a desirable set of liveness and safety security quires is constructed from the GTRBAC constraints. These quires are later used for the model verification process. Their approach is to verify the liveness and safety queries, which is different from us; we use Timed Automata to detect inconsistencies that might be caused due to the interaction between Access Control rules. Another point of difference is that we are dealing with a STRBAC model which is an Access Control model that consider both spatial and temporal constraints, which are required in many application today, while the GTRBAC is an Access Control model that only consider temporal information.

Shafiq et al. [11] show how the various constraints of GTRBAC, such as, Cardinality constraints, SoD constraints, and Role Hierarchy can be modelled using Petri Net. However, identifying inconsistency caused by the interaction between Access control policies is not discussed there. In this paper we demonstrate how we can identify such inconsistencies in the STRBAC model using Timed Automata.

Alloy [5] which is a SAT-solver based has been successfully used for the automatic analysis of Access Control [3], [4], [10]. In [4], [10] Indrakshi et al. present formalization for STRBAC model and they used Alloy for checking Access Control models. In this paper we further the research of [4], [10] by sharing our experience of dealing with inconsistencies identified by automated checkers such as Alloy and Timed Automata. Another effort to verify the consistency of Access Control using Alloy was proposed by [3]. Paper [3] illustrates how GST-RBAC can be analysed using Alloy. In this paper we demonstrate how we could model STRBAC using Timed Automata and then we implement and analyse the Timed Automata models using UPPAAL. A Comparison between the two methods Alloy and Timed Automata is a topic of our future research.

VI. Conclusion and Future work

Using formal methods is always beneficial to model, check and verify computer systems. This paper presents a method that makes use of a formal method to model and check the consistency of STRBAC policies. We have shown that we can translate the STRBAC polices into Timed Automata models, and thus, we can use UPPAAL to implement the produced Timed Automata. To verify the produced Timed Automata we have defined a security query using TCTL, and the model checker UPPAAL has been used to perform the verification.

In the particular, we have used an Access Control example (SECURE Bank System) to illustrate how this methodology works. The translation presented in this paper has been made manually, but our intention is to study if this translation can be made automatically, and in that case to implement a tool supporting this translation.

References

[1] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Hakansson, Paul Petterson, Wang Yi, and Martijn Hendriks. UPPAAL 4.0. In Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems, pages 125126, Riverside, California, USA, September 2006.

[2] Indrakshi Ray and Manachai Toahchoodee. A Spatio-Temporal Access Control Model Supporting Delegation for Pervasive Computing Applications. In Proceedings of the 5th International Conference on Trust, Privacy and Security in Digital Business, pages 48.58, Turin, Italy, September 2008.

[3] Arjmand Samuel, Arif Ghafoor, and Elisa Bertino. A Framework for Specification and Verification of Generalized Spatio-Temporal Role Based Access Control Model. Technical report, Purdue University, February 2007. CERIAS TR 2007-08.

[4] Manachai Toahchoodee and Indrakshi Ray. On the Formal Analysis of a Spatio-Temporal Role-Based Access Control Model. In Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pages 17.32, London, U.K., July 2008.

[5] Jackson.Daniel (2006), Software Abstractions Logic, Language, and Analysis, Cambridge: The Mit Press.

[6] Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A Tutorial on Uppaal. In 4th International School on FormalMethods for the Design of Computer, Communication and Software Systems, pages 200236, Bertinoro, Italy, September 2004.

[7] Liang Chen and Jason Crampton. On Spatio-Temporal Constraints and Inheritance in Role-Based Access Control. In Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, pages 205.216, Tokyo, Japan, March 2008.

[8] J. Bengtsson and W .Yi. Timed automata: Semantics, algorithms and tools. In Lecture Notes on Concurrency and Petri Nets, volume 3098, pages 87124, 2004.

[9] Hsing-Chung Chen, Shiuh-Jeng Wang, Jyh-Horng Wen, Yung-Fa Huang, Chung-Wei Chen: A Generalized Temporal and Spatial Role-Based Access Control Model. JNW 5(8): 912-920 (2010)

[10] Indrakshi Ray and Manachai Toahchoodee. A Spatio-temporal Role-Based Access Control Model. In Proceedings of the 21st Annual IFIPWG11.3Working Conference on Data and Applications Security, pages 211.226, Redondo Beach, CA, July 2007.

[11] Basit Shafiq, James B. D. Joshi, and Arif Ghafoor. Petri-net model for verification of RBAC Policies. Technical report, Purdue University, 2002.

[12] Samrat Mondal, Shamik Sural, and Vijayalakshmi Atluri. Towards Formal Security Analysis of GTRBAC using Timed Automata. In Proceedings of the 14th ACM Symposium on Access control Models and Technologies, pages 3342, Stresa, Italy, June 2009.

[13] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking for real-time systems. In Proc. of the 5th Annual Symposium on Logic in Computer Science, pages 414425. IEEE, 1990.