International Journal of Security and Its Applications
Vol.9, No.8 (2015), pp.23-34
http://dx.doi.org/10.14257/ijsia.2015.9.8.03

Safeness Discussions on TRBAC and GTRBAC Model and an Improved Temporal Role-Based Access Control Model

Meng Liu[1,2] and Xuan Wang[1,3,4,*]

[1]*Computer Application Research Center, Shenzhen Graduate School, Harbin Institute of Technology,*
Shenzhen 518055, China
[2]*School of Mechanical, Electrical and Information Engineering, Shandong University,*
Weihai 264209, China
liumeng@sdu.edu.cn
[3]*Public Service Platform of Mobile Internet Application Security Industry,*
Shenzhen 518055, China
wangxuan@cs.hit.edu.cn
[4]*Shenzhen Applied Technology Engineering Laboratory for Internet Multimedia Application, Shenzhen 518055, China*
**The corresponding author*

Abstract

Bertino et al. propose a temporal Role-based Access Control (TRBAC) model, and Joshi et al. propose a Generalized TRBAC (GTRBAC) model based on TRBAC. Some periodic constraints and duration constraints are introduced to express the corresponding time-based access control policy semantics and enhance the expressiveness of the temporal RBAC model. We have analyzed the TRBAC and GTRBAC models and pointed out that the sufficient conditions for guaranteeing the safeness of the GTRBAC model was not comprehensive, so we have analyzed the reasons and designed a process rule to solve the safety problem. In this paper, an improved process rule is designed to solve the safety problem. In addition, a fault about translating a dependent trigger of TRBAC into an Oracle trigger is analyzed. In order to ensure the temporal RBAC model better, an Improved Generalized Temporal Role-based Access Control (IGTRBAC) based on the TRBAC and GTRBAC models is put forward. The two proposed restrictions in the IGTRBAC model are used to resolve the security problems caused by the dependent trigger and the cardinality constraint on role activation. At last, case study shows that the IGTRBAC model is safe.

Keywords: *Safeness, Temporal Role-based Access Control, constraints, dependent triggers*

1. Introduction

RBAC model has been widely used because it can greatly reduce administrative costs, and meet the most security demands of the current access control policies [1]. The traditional RBAC model is proposed by Sandhu and Ferraiolo *et al.* In 2001 they presented the proposed NIST standard RBAC model [2], and in 2004 this standard was adopted as the American National standard RBAC model [3]. The ANSI RBAC model is already mature and practical. With further research of RBAC, a lot of new security requirements have been proposed and some advances in RBAC model have been considered. Some literatures discuss the automated management of user-role and role-permission assignment and revocation. In Ref. [4] a form of automated management of permissions is represents by automatically adjusting the permission space in a session according to one user's really requirement to perform the ongoing tasks. In the literatures [5-8], the authors achieve the dynamic management and maintenance of privileges according to the user's current spatial position information. These models are very meaningful especially in the mobile network environment. Some advances and enhancement have been discussed based on the traditional RBAC model in the past ten years, the most of which is related to the automation principle of the next-generation RBAC model. In 2008, Sandhu published a paper about the next-generation RBAC [9]. It provides five founding principles for next-generation access control and next-generation RBAC, summarized as ASCAA for Abstraction, Separation, Containment, Automation and Accountability. Sandhu believes that next-generation access control and next-generation RBAC should be based on this expanded set of the five ASCAA principles so as to address real-world protection needs of next-generation systems [9].

In [10, 11], the traditional RBAC model is related to the time factor, so the temporal RBAC model can support temporal constraints. TRBAC [10] introduces temporal constraints on role enabling/disabling to support periodic role enabling and disabling, and temporal dependencies among such actions. Temporal dependencies expressed by means of role triggers can restrain the set of enabling roles and administrators can be free to request necessary role events (actions) to change the enabling or disabling of roles. GTRBAC [11] defines some new temporal constraints to strengthen the expressive power based on TRBAC model, the essential features of which can handle many issues not mentioned in TRBAC model. GTRBAC defines the periodic temporal constraints on user-role and role-permission assignments. And some duration constraints on user-role assignment, role enabling and role-permission assignment are used to specify durations for which these operations is valid. When an event occurs, the duration constraint associated with the event validates the event for the specified duration only. In case no duration constraint exists for the event, the event remains valid until it is disabled by some other means, *e.g.,* by a trigger. The duration constraints on role activation also were introduced that can be classified into two types: total active duration constraint and maximum duration per activation constraint. The total active duration may be specified on per-role and per-user-role basis. The maximum duration constraint per activation can also be specified on a per-role or per-user role basis. The number of concurrent activations of a role can also be controlled by the cardinality restriction on role activation, which can be categorized into two types: total *n* activations constraint and maximum *n* concurrent activations constraint. Temporal role hierarchies and separation of duty constraints are not addressed in [10], so they are discussed and introduced in GTRBAC [11]. However, some safety state could be inconsistent so that a priority was given to these actions to solve conflicts, such as the simultaneous enabling and disabling of a role. As expected, the action with the highest priority is executed in TRBAC model. And a safeness condition was introduced to be verified in polynomial time and to guarantee that a given Role Enabling Base (REB) should have one and exactly one execution model. It could use the notion of labeled dependency graph to analyze the safeness of the execution model. Joshi applies these definitions directly to the GTRBAC model, which is used to guarantee the safety and consistent of GTRBAC model. Ref. [12] has analyzed the sufficient condition for the safeness defined in TRBAC and GTRBAC model, and pointed out that the sufficient condition definition applied directly to GTRBAC model was not comprehensive. So some approaches to guarantee the safeness of GTRBAC model have been propose in Ref. [12], but TRBAC and GTRBAC model still have some others problems to be solved.

The main contributions of this paper can be summarized as follows:

(1) Continue to analyze the sufficient condition for the safeness defined in TRBAC and GTRBAC model based on some previous research results in Ref.

[12], and then an improved process rule is designed to solve the safety problem.

(2) Analyze and correct a critical fault about translating a role triggers to Oracle triggers in TRBAC model.

(3) Propose an improved GTRBAC model based on TRBAC and GTRBAC model, which does not require any conditions to guarantee safeness and the IGTRBAC must have exactly one execution model.

2. Preliminaries

In this section we first introduce the RBAC model. Then we describe the formal definition of periodic time, which is very important in TRBAC and GTRBAC model.

2.1. RBAC Model

Sandhu and Ferraiolo proposed the original traditional RBAC model [1-2]. The ANSI RBAC [3] model and its components are shown in Fig. 1. The ANSI RBAC model consists of four model components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations and Dynamic Separation of Duty Relations. Core RBAC includes sets of five basic data elements called users, roles, objects, operations, and permissions. Hierarchical RBAC introduces role hierarchies, which are a key aspect of RBAC model. Separation of Duty (SoD) relations includes Static Separation of Duty Relations and Dynamic Separation of Duty Relations. They are used to enforce conflict of interest policies to prevent users from exceeding a reasonable level of authority for their tasks. More information can be found in [1-3].

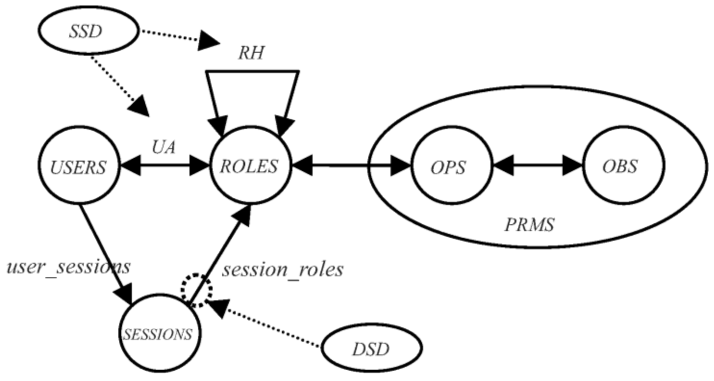

Figure 1. ANSI RBAC Model

2.2. Periodic Expression

The concepts of the periodic expression and periodic time are taken from [10-11,13]. Periodic time is defined through a symbolic formalism as a tuple (I, P), where P is a periodic expression denoting an infinite set of periodic time instants, and I =[begin, end], I is a time interval denoting the lower and upper bounds for all the instants in P. Periodic time uses the notion of calendar defined as a countable set of contiguous intervals. We assume a set of calendars containing the calendars Hours, Days, Weeks, Months, and Years, where Hours is the calendar with the finest granularity. Given two calendars C_1 and C_2, C_1 is said to be a sub-calendar of C_2, written as $C_1 \subseteq C_2$, if C_2 consists of a finite number of intervals of C_1. For example, C_1 represents Hour and C_2 represents Day, and one day consists of 24 hours, so Hour is a sub-calendar of Day. Calendars can be combined to represent more general periodic expressions denoting periodic intervals, such as the set of Mondays, the set of the tenth hour of every day, or the set of the third hour of the first day of each month. A periodic expression is defined as:

$$P = \sum_{i=1}^{n} O_i \cdot C_i \triangleright r \cdot C_d \qquad (1)$$

where $C_d, C_1, C_2, \cdots, C_n$ are calendars, $O_1 = all$, $O_i \in 2^N \cup \{all\}$, $Ci \subseteq C_{i-1}$ for $i = 2, \cdots, n$, $C_d \subseteq C_n$, $r \in 2^N$. The symbol $\triangleright$ separates the first part of the periodic expression that denotes the set of starting points of the intervals from the second part that denotes the duration of each interval in terms of calendar C_d. For example, $all \cdot Years + \{3,7\} \cdot Months \triangleright 2 \cdot Months$ represents the set of intervals starting at the same instant as the third and seventh month of each year and having a duration of two months, i.e. March, April, July and August of each year. O_i can be ignored when its value is all, whereas it is represented by its unique element when it is a singleton. Similarly, $r \cdot C_d$ is omitted when it is equal to $1 \cdot C_n$. The set of intervals corresponding to a periodic expression P is defined as $\Pi(P)$. Similarly, the set of time instants corresponding the periodic time (I, P) is defined as $Sol(I, P)$. The notion of periodic time is used to both of the TRBAC and GTRBAC model and more detailed information can refer to [10-11,13].

3. Discussion on Sufficient Safeness Condition for TRBAC and GTRBAC Model

Some previous research results will be introduced again in this section because Ref. [12] introduced them in Chinese.

A set of elements of the Periodic events on role enabling/disabling and role triggers is called a Role Enabling Base (REB) in TRBAC model [10], expressed as R. And a set of all the event expressions, constraints, and triggers in a GTRBAC system is called a Temporal Constraint and Activation Base (TCAB), expressed as Γ.

The security of TRBAC and GTRBAC is defined from the point of its execution model and semantics. The following definition and theorem about safeness conditions are proposed to ensure a given REB in TRBAC model or a given TCAB in GTRBAC model to be safe.

Definition 1. (Safeness [10]) A REB R is safe if its dependency graph DG_R contains no cycles in which some edge is labeled "-".

Theorem 1. (See [10].)If a REB R is safe, then for all request streams RQ, there exists exactly one canonical execution model of R and RQ.

Joshi applies directly the definitions and theorems in TRBAC to GTRBAC model to illustrate its safety. In GTRBAC model, various types of conflicts are defined, including conflicts between events of the same category, conflicts between events of different categories and conflicts between constraints. Unambiguous semantics can arise because of these conflicts. So Joshi defines three classes of treatment of conflict to ensure semantic disambiguation. And GTRBAC can be safety and consistency because of the treatment of conflict. However, it is not correct that Joshi applies directly the definition 1 and theorem 1 in the GTRBAC model. The following example is used to analyze the problem.

Example 1. Consider a TCAB Γ of a GTRBAC model shown in Table 1[12].

Table 1. A an Example of a TCAB Γ

Enable DayDoctor $\rightarrow$ Enable DayNurse
Disable DayDoctor $\rightarrow$ disable DayNurse
Enable NightDoctor $\rightarrow$ Enable NightNurse
Disable NightDoctor $\rightarrow$ disable NightNurse
Activate DayNurse for Elizabeth $\rightarrow$ Enable NurseInTraining

Copyright ⓒ 2015 SERSC

Deactivate DayNurse for Elizabeth $\rightarrow$ disable NurseInTraining
(1 active $_{R\_con}$ DayNurse)

$RQ(t)$ = { enable DayDoctor, Activate DayNurse for Elizabeth, Activate DayNurse for Rose } . For the sake of this discussion, we just consider the security state of the roles. The constraint of "(1 active $_{R\_con}$ DayNurse)" means that the number of concurrent activations role DayNurse is up to 1.

(1) If real-time activation request "Activate DayNurse for Elizabeth" processed first, then $EV(t)$ = { enable DayDoctor, Enable DayNurse, Activate DayNurse for E lizabeth } , and the valid role set is { DayDoctor, DayNurse, NurseInTraining } at time t .

(2) If real-time activation request "Activate DayNurse for Rose" processed first, then $EV(t)$ = { enable DayDoctor, Enable DayNurse, Activate DayNurse for Rose } , and the valid role set is { DayDoctor, DayNurse } at time t .

This example has two execution models and it is not safe. But according to the detection algorithm [10-11], the example should be safe. So given a TCAB satisfied safeness condition defined in definition 1 in GTRBAC, it cannot guarantee the safety of the model. We know that Joshi expands the TRBAC model and introduces the temporal role activation events in GTRBAC model. And these role activation events also may trigger other management events, such as enabling or disabling role event in example 1. But GTRBAC model does not clearly define how to deal with the role activation event conflicts when the constraints of role activation are violated. We can define an unambiguous process priority for every role activation event at time t . These activation events can be processed according to its priority and the constraints of role activation, and the safety problem can be solved. The previous example is discussed by the maximum number of concurrent role activations constraint of per-role. In fact, the other cardinality constraints on role activation include per-role total number activations constraint, per-user-role total number activations constraint and per-user-role maximum number concurrent activations constraint and all of them can cause the same problem. In this paper, we define a simple unambiguous priority mechanism to serialize the role activation events at time t . Even though all the role activation events happen at the same time t , from the micro perspective, they really are not the same, i.e., these events happen in sequential order. Suppose that all the role activation events can be inserted into a database table with an auto-increment field. When finished inserting, every role activation event at time t can have a unique serial number. The role activation event with the minimum number would happen first and so on. Based on the serial number defined with every activation event, an improved rule based the rule in Ref. [12] can be defined to deal with the role activation event conflicts when the cardinality constraints of role activation are violated.

Definition 2. Consider a set of role activation events at time t , $AS_t = \{E_1, E_2, ..., E_n\}$, Every $E_i \in AS_t$ is denoted by (s :activate r for u), where i is a serial number of a role activation event, s is the session, r is the activation role and u is the user in session s . And the N_{r_active} is the remaining total number of role r activation, the N_{r_max} is the remaining concurrent number of role r activations, the $N_{u\_r\_active}$ is the remaining total number of role r activation by user u and the $N_{u\_r\_max}$ is the remaining concurrent number of role r activations by user u .

```
for each $E_i$ in $AS_t$ do
    $E_i$ =( $s$ :activate $r$ for $u$ )
```

```
  if $N_{r\_active}-1>0$ and $N_{r\_max}-1>0$ and $N_{u\_r\_active}-1>0$ and $N_{u\_r\_max}-1>0$ then
        user $u$ active $r$ in session $s$ ;
        $N_{r\_active}--$;
        $N_{r\_max}--$;
        $N_{u\_r\_active}--$;
        $N_{u\_r\_max}--$;
     end
end
```

Role activation events, “Activate DayNurse for Elizabeth" and “Activate DayNurse for Rose" can be numbered in the order, but no matter what a high priority, example 1 has only one execution model according to Definition 2 for conflict resolution. Therefore, while applying Definition 2 to eliminate the conflicts because the role activation events violation the cardinality constraints, it is correct that theorem 1 is applied to the GTRBAC model.

Theorem 2. For all request streams RQ and role activation events set $AS_t \subset RQ(t)$, if A TCAB Γ is safe and AS_t can be handled according to definition 2, then there exists exactly one canonical execution model of Γ and RQ.

Based on the previous discussion, the correctness of the theorem 2 is obvious.

4. Discussion on TRBAC Trigger Translation Algorithm

In fact, there remain some problems in the system implementing TRBAC based on an Oracle DBMS while the Trigger Compiler translates a role trigger into Oracle trigger.

Example 2. Consider a REB R consisting of the role triggers,

1. enable R_1 $\rightarrow$ Enable R_2
2. enable R_0 $\rightarrow$ disable R_1

The REB R meets the safeness condition in definition 1 and should have exactly one canonical execution model. Suppose that $RQ(t)=\{$ enable R_1 , enable R_0 $\}$, $ST(t-1)=\varnothing$. According to the syntax and semantics of the TRBAC model, there are the possibilities:

1. Fire the second trigger; this yields: $EV(t)=\{$enable R_1, enable R_0, disable R_1 $\}$. In this case enable R_0 is not blocked, while enable R_1 is. Intuitively, the second trigger blocks the first one. So $ST(t)=\{R_0\}$.
2. If trigger 1 is evaluated before trigger 2, its firing generates Enable R_2 and $R_2 \in ST(t)$, and this is not correct since the subsequent evaluation of trigger 2 generates disable R_1 which should have blocked the firing of trigger 1.

To cope with this problem, Bertino et al. impose some restrictions on the specification language supported by the current TRBAC prototype that ensure the correct evaluation of triggers even in the absence of a priority mechanism when implementing TRBAC on top of an Oracle DBMS. The first restriction they impose is that actions caused by the firing of a trigger must all have the same priority. Additionally, to ensure that the evaluation of triggers with the same priority always enacts the correct semantics, they impose a further restriction to triggers causing instantaneous actions (i.e., those with $\Delta t=0$, also called instantaneous triggers): if an event enable R appears in the head of an instantaneous trigger, then the body must contain $\neg$enabled R, and if the event disable R appears in the head of an instantaneous trigger, then the body must contain enabled R [10].

So the two role triggers in example 2 should be modified as follows.

1. Enable R_1, $\neg$enabled R_2 $\rightarrow$ Enable R_2

2. Enable R_0, enabled R_1 $\rightarrow$ disable R_1.

Suppose that $RQ(t) = \{$ enable R_1, enable R_0 $\}$, $ST(t-1) =$. This yields: $EV(t) = \{$enable R_1, enable R_0, enable R_2 $\}$ and $ST(t) = \{R_0, R_1, R_2\}$. As can be seen, after adding restrictions the Example 2 still has exactly one canonical execution model, but the execution model of the model have changed.

Furthermore, Ref. [10] illustrates how the *Trigger Compiler* translates role triggers into Oracle triggers. The core idea is to use a "before insert" trigger of a data table to implement dependent role trigger. But Bertino et al. do not account for a fault that an action caused by a role trigger and inserted into table *Actions* or *Deferred_Actions* also can cause another role trigger and insert a next new action into table *Actions* or *Deferred_Actions*. So the *Trigger Compiler* to translate a role trigger into an Oracle trigger must be improved to ensure the integrity. We show an example to illustrate this case.

Example 3. Consider a REB R consisting of the role triggers,

1. disable DayDoctor $\rightarrow$ disable DayNurse
2. disable DayNurse $\rightarrow$ disable NurseInTraining.

According to the *Trigger Compiler* in Ref. [10], the Oracle triggers are defined as follows.

```
create trigger RT1
before insert on Events
for each row
when (new.action = ‘disable’ AND new.role = ‘DayDoctor’)
declare
X, Y number;
begin
select count (*) into X from Enabled_Roles where role = ‘DayDoctor’;
if X > 0 then
select count (*) into Y from Actions where (role = ‘DayDoctor’ AND action = ‘enable’ AND
priority > 0);
if Y = 0 then
insert into Actions values (‘DayNurse’, ‘disable’, all, 0);
endif
endif
end;
create trigger RT2
before insert on Events
for each row
when (new.action = ‘disable’ AND new.role = ‘DayNurse’)
declare
X, Y number;
begin
select count (*) into X from Enabled_Roles where role = ‘DayNurse’;
if > 0 then
select count (*) into Y from Actions where (role = ‘DayNurse’ AND action = ‘enable’ AND
priority > 0);
if Y = 0 then
insert into Actions values (‘NurseInTraining’, ‘Disable’, all, 0);
endif
endif
end;
```

The first role trigger in example 3 can be translated to the Oracle Trigger RT1 and the second one can be translated to the Oracle Trigger RT2. Suppose that $RQ(t) = \{$disable

DayDoctor }, $ST(t-1)=\{$ DayDoctor, DayNurse, NurseInTraining }. At time t Oracle Trigger RT1 can be triggered and it will execute an insert operation of "insert into Actions values ('DayNurse', 'disable', all, 0);". But Oracle Trigger RT2 cannot be triggered because the Oracle Trigger RT2 is with "before insert on Events" but not "before insert on Actions". So $ST(t)=\{$ NurseInTraining }. And the state of role nurseInTraining is not correct at time t. In this paper, we only discuss the problem and the necessity to improve the *Trigger Compiler*.

5. IGTRBAC Model

The safeness of TRBAC and GTRBAC models is discussed from the point of the triggers. If the role triggers can satisfy the safeness condition, the TRBAC and GTRBAC model are safe. Bertino and Joshi point that the safeness of the triggers is a sufficient condition. Necessary conditions are much harder to find and of little practical help. In general, checking model existence and model uniqueness are NP-hard problems. So they have not given a necessary condition to ensure the model security. Model uniqueness is the sufficient condition of the model security and the model security also is the sufficient condition of model uniqueness. If a REB R is not safe and there is one streams RQ and there exists exactly one canonical execution model of R and RQ, then we also can say that the model is safe.

Example 4. Consider a REB $R=\{$ enable $R \rightarrow$ disable S, enable $S \rightarrow$ disable R } And $RQ(t)$ = { enable R }, $RQ(t+1)$ = { enable S }, then,

1. $EV(t)$ = { enable R, disable S }, $ST(t)$ = { R }.
2. $EV(t+1)$ = { enable S, disable R }, $ST(t+1)$ = { S }.

Though the REB is not safe, this example has exactly one execution model. So we can say that it is safe and the request stream also can influence the security of a model. But we cannot find all the safe request streams. Whether we can design a temporal RBAC and it is safe and has exactly one canonical execution model for all request streams RQ even if the REB is not safe? So we propose an improved model based on GTRBAC, called Improved Generalized Temporal Role-based Access Control (IGTRBAC). The improved idea of IGTRBAC is simple. We impose two restrictions on GRTBAC model. They can be expressed as follows:

1. Add the definition 2 as a conflict treatment rule into GTRBAC model to ensure the safeness caused by the cardinality constraints on role activation events.
2. Define the trigger in GTRBAC model to be a deferred trigger and not an instantaneous trigger, i.e., $\Delta t > 0$ for an action in a head of a trigger.

The first restriction has been discussed in section 3, and we will discuss the second restriction.

Example 5. Consider a TCAB Γ = {enable $R \rightarrow$ disable S after 1, enable $S \rightarrow$ disable R after 1 } and $RQ(t)$ = {enable R, enable S }, then

1. $EV(t)$ = { enable R, enable S }, $ST(t)$ = $\{R,S\}$.
2. $EV(t+1)$ = { disable S, disable R }, $ST(t+1)$ $=\varnothing$.

According to the discussing in Ref. [10], if the two role triggers are instantaneous trigger, the TCAB has no exactly one canonical execution model and is not safe. Obviously, after the former instantaneous trigger is changed to a deferred trigger, the TCAB has exactly one canonical execution model and is safe, i.e., we can eliminate the cycle of the dependency graph DG_R in which some edge is labeled "-" through deferred action. So we don't even have to think about the dependency graph and verify the safe condition of a TCAB.

Theorem 3. IGTRBAC model has exactly one canonical execution model.

According to the previous discussion, the correctness of the theorem 3 is obvious, and thus this article does not give proof.

Now there is a new problem: Whether imposing the restriction of $\Delta t > 0$ will reduce the expression power of GTRBAC model. For a former instantaneous trigger, if the deferred time is small enough and the deferred action can be trigger as soon as possible, the execute state of an access control system will not be influenced from the point of practical application. We use two examples to demonstrate the fact result and significance.

First, we discuss the example 5 again. It has exactly one canonical execution model and is safe. If the two role triggers are instantaneous trigger, it means that the valid states of role R and S are mutually exclusive. According to our second restriction, if Δt is small enough, we can think that $ST(t)$ = $\{R,S\}$ is changed to $ST(t)$ $=\varnothing$, which means that the role R and S are not enabled at time t.

Example 6. Consider a TCAB Γ = {disable $R \rightarrow$enable S after 1, disable $S \rightarrow$ enable R after 1 } and $RQ(t)$ = {disable R, disable S }, then

1. $EV(t)$ = {disable R, disable S }, $ST(t)$ $=\varnothing$.
2. $EV(t+1)$ = {enable S, enable R }, $ST(t+1)$ = { R, S }.

This example also has exactly one canonical execution model and is safe. But we can think that this example may cause a problem of system security. When the two role triggers are instantaneous triggers, the TCAB has no execution model and the states of role R and S should not be changed. In other words, example 6 with deferred time does not have a same execution model as the former. In fact, there are two mutually exclusive triggers at time t and it means that $ST(t)$ ={ R,S } rather than $ST(t)$ = at time t. So $ST(t+1)$ = { R,S } is reasonable.

Example 7. Consider a REB R consisting of the role triggers,

1. enable R_1 $\rightarrow$ enable R_2 after 1
2. enable R_0 $\rightarrow$ disable R_1 after 1

The REB R also should have exactly one canonical execution model. Suppose that $RQ(t) =$ { enable R_1, enable R_0 }, $ST(t-1) =$.

1. $EV(t)$ = {enable R_1, enable R_0 }, $ST(t) = \{R_1, R_0\}$.
2. $EV(t+1)$ = {enable R_2, disable R_1 }, $ST(t+1) = \{R_2, R_0\}$.

We can see that Example 7 with deferred time has not the same execution model as example 2. Though the execution model has changed, the priority of trigger can be ignored. It can simplify the implementation of the prototype which has been discussed in section 4.

6. Conclusions

GTRBAC model extends some time constraints and event expression based on TRBAC model and can provide a more flexible security features. Meanwhile GTRBAC model introduces the sufficient condition and polynomial time detection algorithm to ensure its safety. But GTRBAC model does not clearly define how to deal with the role activation event conflicts when the constraints of role activation are violated. So we can design a process rule to solve the safety problem. Then the sufficient safeness condition of TRBAC can be applied into GTRBAC model. We also discuss the problem existed in the *Trigger Compiler* in TRBAC model and the necessity to improve the *Trigger Compiler* to ensure the correct implementation of the TRBAC prototype. To ensure the safety of GTRBAC better, we propose an Improved Generalized Temporal Role-based Access Control by imposing two restrictions on GTRBAC model. The new model has exactly one canonical execution model and is safe for all request streams RQ even if the

TCAB is not safe. And it also can simplify the implementation of the prototype. Though IGTRBAC has exactly one canonical execution model and is safe, it is also very important that whether the System Trace of an execution model is consistent with the expectations and how to check the System Trace. Both of GTRBAC and GTRBAC do not consider this problem. Our future work is to explore the methods and solve this problem.

Acknowledgements

This work was supported by the Public Service Platform of Mobile Internet Application Security Industry under Grants Shenzhen Development and Reform Commission No.2012720, Shenzhen Applied Technology Engineering Laboratory for Internet Multimedia Application under Grants Shenzhen Development and Reform Commission No.2012720, Research on Key Technology in Developing Mobile Internet Intelligent Terminal Application Middleware under Grants No.JC201104210032A.

References

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, “Role-based access control models”, Computer, vol. 29, no. 2, (**1996**), pp. 38-47.
[2] 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 (TISSEC), vol. 4, no. 3, (**2001**), pp. 224-274.
[3] ANSI, Role based access control, ANSI/INCITS 359-2004 (**2004**).
[4] J. R. Mühlbacher and C. Praher, “Ds rbac-dynamic sessions in role based access control”, J. UCS, vol. 15, no. 3, (**2009**), pp. 538-554.
[5] F. Hansen and V. Oleshchuk, “Srbac: A spatial role-based access control model for mobile systems”, in: Proceedings of the 7th Nordic Workshop on Secure IT Systems (NORDSEC’03), Citeseer, (**2003)**, pp. 129-141.
[6] E. Bertino, B. Catania, M. L. Damiani and P. Perlasca, “Geo-rbac: a spatially aware rbac”, in: Proceedings of the tenth ACM symposium on Access control models and technologies, ACM, Stockholm, Sweden, (**2005**), pp. 29-37.
[7] M. L. Damiani, E. Bertino and P. Perlasca, “Data security in location-aware applications: an approach based on rbac”, International Journal of Information and Computer Security, vol. 1, no. 1/2, (**2007**), pp. 5-38.
[8] Y.-G. Kim and J. Lim, “Dynamic activation of role on rbac for ubiquitous applications”, in: Proceedings of the 2007 International Conference on Convergence Information Technology, IEEE Computer Society, Washington, DC, USA, (**2007)**, pp. 1148-1153.
[9] R. Sandhu and V. Bhamidipati, “The ascaa principles for next-generation role-based access control”, in: Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES), IEEE Computer Society, Barcelona, Spain, (**2008)**.
[10] E. Bertino, P. A. Bonatti and E. Ferrari, “Trbac: A temporal role-based access control model”, ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, (**2001**), pp. 191-233.
[11] J. B. D. Joshi, E. Bertino, U. Latif and A. Ghafoor, “A generalized temporal role-based access control model”, IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 1, (**2005**), pp. 4-23.
[12] M. Liu and X. Wang, “Security analysis on gtrbac model and its improvement”, Computer Applications and Software, vol. 29, vol. 10, (**2012**), pp. 300-303 (in Chinese).
[13] E. Bertino, C. Bettini, E. Ferrari and P. Samarati, “An access control model supporting periodicity constraints and temporal reasoning”, ACM Transactions on Database Systems (TODS), vol. 23, no. 3, (**1998**), pp. 231-285.

Authors

Meng Liu, he received his Master’s degree in Computer Sciences from the School of Computer Science and Technology, Shandong University, China, in 2004. Currently, he is working toward the Ph.D. degree in Computer Science at the Computer Application Research Center, Harbin Institute of Technology Shenzhen Graduate School, China and is a lecturer in the School of Mechanical, Electrical and

Information Engineering, Shandong University, Weihai, China. His main research interests include network and information security.

Xuan Wang, he received his M.S. and Ph.D. degrees in Computer Sciences from Harbin Institute of Technology, Harbin, China, in 1994 and 1997 respectively. He is a professor and Ph.D. supervisor in the Computer Application Research Center, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China. His main research interests include artificial intelligence, computer vision, computer network security and computational linguistics. He is a member of the IEEE.

Copyright ⓒ 2015 SERSC