Proceedings of International Workshop on Internet and Distributed Computing Systems (IDCS' 08)
24 December, 2008, Khulna, Bangladesh

# A History Based Semantic Aware Access Control Model Using Logical Time

Ali Noorollahi Ravari, Morteza Amini, Rasool Jalili, and Jafar Haadi Jafarian
*Network Security Center, Computer Engineering Department,*
*Sharif University Of Technology, Tehran, Iran*
*noorollahi@ce.sharif.edu,jafarian@ce.sharif.edu, m_amini@ce.sharif.edu, jalili@sharif.edu*

## Abstract

With the advent of semantic technology, access control cannot be done in a safe way unless the access decision takes into account the semantic relationships among the entities in a semantic-aware environment. The SBAC model (Semantic Based Access Control model) considers this issue in its decision making process. However, time plays a crucial role in new computing environments, which is not supported in SBAC. In this paper, we propose the Temporal Semantic Based Access Control (TSBAC) model, as an extension of SBAC, which enhances the specification of user-defined authorization rules by constraining time interval and temporal expression over users' history of accesses. TSBAC uses logical time, rather than to real time, in its authorization rules. A formal semantics for temporal authorizations is provided and conflicting situations (due to the semantic relations of the SBAC model and a sub-interval relation between authorizations) are investigated and resolved in our proposed model. An architecture for the access control system based on TSBAC is presented.

## 1. Introduction

An important requirement of any information management system is to protect data and resources against unauthorized disclosure (confidentiality) and unauthorized or improper modifications (integrity), while at the same time ensuring their availability to legitimate users (no denials-of-service). Therefore, enforcing protection requires that every access to a system and its resources be controlled and all and only authorized accesses can take place [1]. The development of an access control system requires the definition of the regulations according to which access is to be controlled and their implementation as functions executable by a computer system.

An important requirement, common to many applications, is related to the temporal dimension of access permissions. In these systems, permissions are granted based on previous authorizations given to the users of the system or denied in specific time points (in the past). Another critical requirement is the possibility of expressing the semantic relationships that usually exist among different authorization elements, i.e. subjects, objects, and actions. To overcome this challenge, our model is constructed based on the SBAC model (Semantic Based Access Control model) [2, 3] which is a semantic-based access control model. SBAC authorizes users based on the credentials they offer when requesting an access right. Ontologies are used for modeling entities along with their semantic interrelations in three domains of access control, namely subjects domain, objects domain, and actions domain. To facilitate the propagation of policies in these three domains, different semantic interrelations can be reduced to the subsumption relation.

In this paper we unify the two concepts mentioned above; that is, we use the SBAC model (as the base model), and associate a temporal expression with each authorization. Thus, in this paper we employ logical time operators to specify historical constraints over the elements of the history base in authorization rules. Furthermore, a time interval bounds the scope of applicability of the temporal authorization rules, and usage of elements of history (e.g., $[1, 2, 20]$ shows that the authorization is valid for time interval starting at '1' and ending at '20', and history elements with the time stamp greater than 2 are taken into account). Thus, the main feature provided by our model is the possibility of specifying authorization rules which express temporal dependencies among access events in the past (stored in *a history base* in the form of $done(t, s, o, a)$ and $denied(t, s, o, a)$)

The rest of this paper is structured as follows: Section 2 discusses the related work in this topic. Section 3 gives a brief introduction of the SBAC model and describes the model of time used throughout our work. In section 4, we represent our authorization rules in detail, offer their formal semantics, and conflict resolution in access decision point. Section 5 gives an architecture for the access control system based on the proposed model. In Section 6 an implementation of the basic components of the model using CLIPS is presented In section 7, a case study in a banking environment is described. A qualitative and quantitative evaluation of the model is discussed in section 8. Finally, Section 9 concludes the paper and gives some future works on the topic.

## 2. Related Work

Access control systems for protecting web resources along with credential based approaches for authenticating users have been studied in recent years. With the advent of Semantic Web, new security challenges were imposed to security systems. Bonatti *et al.* [4] have discussed open issues in the area of policy specification for Semantic Web community such as important requirements for access control policies. Developing security annotations to describe security requirements and capabilities of web service providers and requesting agents have been addressed by Rabitti *et al.* [5]. Qin and Atluri [6] proposed a concept level access control model that considers some semantic relationships in the level of concepts in objects domain. SBAC which is the basis of our model, is proposed by Javanmardi *et al.* [2, 3], and is based on the OWL ontology language and considers the semantic relationships in the domains of subjects, objects, and actions to make decision about an access request.

The first security policy based on the history of events was introduced as Chinese Wall Security Policy (CWSP) [7]. The objective of CWSP is to prevent information flows which cause conflict of interest for individual consultants. Execution history also plays a role in Schneider's security automata [8] and in the Deeds system of Edjlali [9]. However, such works focus on collecting a selective history of sensitive access requests and use this information to constrain further access requests; for instance, network access may be explicitly forbidden after reading certain files. Another approach that considers the history of control transfers, rather than a history of sensitive requests, is presented by Abadi and Fournet [10].

In a basic authorization model, an authorization is modeled by a triple $(s, o, \pm a)$, interpreted as "subject *s* is (not) authorized to exercise access right *a* on object *o*". Recently, several extensions to this basic authorization model have been suggested. One of them is the temporal extension, which increases the expressive power of the basic authorization model [11-15]. In the model proposed by Bertino *et al.* [11], an authorization is specified as $(time, auth)$, where $time = (t_b, t_e)$ is the time interval in which the authorization $auth = (s, o, m, pn, g)$ is valid. In *auth, s* represents the subject, *o* the object, and *m* the privilege, *pn* is a binary parameter indicating whether an authorization is negative or positive, and *g* represents the grantor of the authorization. This model also allows operations *WHENEVER*, *ASLONGAS*, *WHENEVERNOT*, and *UNLESS* on authorizations. For example, *WHENEVER* can be used to express that a subject $s_i$ can gain privilege on object *o* whenever another subject $s_j$ has the same privilege on *o*. Later Bertino *et al.* [14] extended the temporal authorization model to support periodic authorizations. They completed their research in [16] by presenting a powerful authorization mechanism that provides support for: (1) periodic authorizations (both positive and negative), that is, authorizations that hold only in specific periods of time; (2) user-defined deductive temporal rules, by which new authorizations can be derived from those explicitly specified; (3) a hierarchical organization of subjects and objects, supporting a more adequate representation of their semantics. From the authorizations explicitly specified, additional authorizations are automatically derived by the system based on the defined hierarchies.

## 3. Preliminaries

In this section we give a brief introduction to the SBAC model, proposed by Javanmardi *et al.* [2, 3], and introduce the model of time used throughout our work.

### 3.1 Introduction to SBAC

Fundamentally, SBAC consists of three basic components: Ontology Base, Authorization Base, and Operations. Ontology Base is a set of ontologies: Subjects–Ontology (SO), Objects–Ontology (OO), and Actions–Ontology (AO). By modeling the access control domains using ontologies, SBAC aims at considering semantic relationships in different levels of ontology to perform inferences to make decision about an access request. Authorization Base is a set of authorization rules in the form of $(s, o, \pm a)$ in which $s$ is an entity in SO, *o* is an entity defined in OO, and *a* is an action defined in AO. In the other words, a rule determines whether a subject which presents a credential $s$ can have the access right $a$ on object $o$ or not.

The main feature of the model is reduction of semantic relationships in ontologies to subsumption relation. Given two concepts $C$ and $D$ and a knowledge base $\Sigma$, $C \prec D$ denotes that *D* subsumes *C* in $\Sigma$. This reasoning based on subsumption represents that *D* (the subsumer) is more general than *C* (the subsumee).

### 3.2 Modeling of Time

We assume that the system is composed of a single process, and we timestamp each event with a counter based clock. The clock ticks every time an event occurs.

## 4. SBAC with Logical Time Constraints

In some applications, only the logical sequence of events is of considerable importance, or due to the system specifications, time stamping of events with real time is impossible. Thus, in these situations, we use logical time instead of real time and timestamp events using the logical time scheme introduced in section 3.2.

**Definition (Temporal Authorization Rule)** A temporal authorization rule is a triple $([t_{sv}, t_{sh}, t_f], (s, o, \pm a), F)$, where $t_{sv}, t_{sh}, t_f \in LogicalTime,$ , and $t_{sv}, t_{sh} \leq t_f$. In this notation, $[t_{sv}, t_{sh}, t_f]$ represents the authorization $(s, o, \pm a)$ validation time interval, and formula $F$ is a temporal constraint which is formally defined in Table 1.

**Table 1. Definition of temporal predicate *F***

$$A ::= done(s,o,a)|denied(s,o,a)|$$
$$\neg done(s,o,a)|\neg denied(s,o,a)$$
$$E ::= prev(A)|past\#(A)|H(A)|sb\#(A,A)|$$
$$ab(A,A)|ss(A)|during(A,A)$$
$$F ::= true|false|E|\neg E|E \wedge E|E \vee E|E \rightarrow E|E \leftrightarrow E$$

The temporal authorization rule $([t_{sv}, t_{sh}, t_f], (s, o, \pm a), F)$ states that subject presenting credential *s* is allowed (or not allowed) to exercise access *a* on object *o*, in the time interval $[t_{sv}, t_f]$ if formula F is evaluated to true by the access events occurred in the time interval $[t_{sh}, t_f]$.

**Definition (Temporal Authorization Base)** A temporal authorization base (TAB) is a set of temporal authorization rules in the form of $([t_{sv}, t_{sh}, t_f], (s, o, \pm a), F)$.

**Definition (History Base)** A History Base is a set of authorizations with timestamp, in the form of $done(t, s, o, a)$ that means access *a* has been granted to subject presenting credential *s* on object *o* at logical time *t*, and $denied(t, s, o, a)$ that means the system has denied access *a* on object *o* at logical time *t* requested by subject presenting credential *s*.

## 4.1 Informal Meaning of Temporal Authorization Rules

The intuitive meaning of (logical time) temporal authorization rules is as follows (in these statements *auth* is used instead of $(s, o, \pm a)$):

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, prev(A)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if *A* is evaluated to true at the time $(t-1)$, and $t - 1 \geq t_{sh}$.
$\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, past\#(A)}\right)$: Authorization *auth* is valid at the time of request *(t)*, $t_{sv} \leq t \leq t_f$, if *A* is evaluated to true # of times in the time interval $[t_{sh}, t]$ in the history base.

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, H(A)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if *A* is evaluated to true at all time-points in the time interval $[t_{sh}, t]$ in the history base.

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, sb\#(A_1, A_2)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if $A_1$ is evaluated to true # of times before the last occurrence of $A_2$ in the time interval $[t_{sh}, t]$.

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, ab(A_1, A_2)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if $A_1$ is evaluated to true at $t' (t_{sv} \leq t' \leq t)$, and there exist a time-point $t"$ $(t' \leq t" \leq t)$, in which $A_2$ is evaluated to true.

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, ss(A_1, A_2)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if $A_1$ is always evaluated to true, from the first occurrence of $A_2$ in the time interval $[t_{sh}, t]$ till *t*.

– $\left([\boldsymbol{t_{sv}, t_{sh}, t_f}], \boldsymbol{auth, during(A_1, A_2)}\right)$: Authorization *auth* is valid at the time of request (*t*), $t_{sv} \leq t \leq t_f$, if $A_1$ is not true before the first, or after the last time-point in which $A_2$ is true in the time interval $[t_{sh}, t]$.

## 4.2 Comparison with Linear Time Temporal Logic Operators

As mentioned in previous sections, TSBAC is an access control model that makes its access control decision based on the temporal relation between users' access events in the history. Due to this requirement, we considered a modified set of operators of the Propositional Linear Temporal Logic (Future and Past version) [17]. In addition, some of the operators were added due to the requirements of real environments where the access control system based on the model is applicable. In the following, an overall comparison of the operators of TSBAC and LTL is presented. The newly added operators are described next.

The basic temporal operators of this system are '*Fp*' (sometime *p*or eventually *p*), '*Gp*' (always *p* or henceforth *p*), '*Xp*' (next time *p*), '*p U q*' (*p* until *q*), and the modality '*p B q*' (*p* precedes *q*).

*'Prev'* in TSBAC is the past time equivalent of '*X*' in LTL (or $X^{-}$), '*H*' is the past time equivalent of '*G*' (or $G^{-}$), '*sb#*' is the past time equivalent of '*B*' which takes into account the number of instants in which the first operand ($A_1$) is evaluated to true. In some situations we need to identify the situation in which an event ($A_1$) has been repeated in all timepoints in the past. Thus, the operator '*ab*' seems necessary. '*ss*' is the '*Since*' operator but with a minor modification; in addition to being a past time equivalent, it is required that the first operand ($A_1$) holds from the moment that $A_2$ is evaluated to true.

In order to describe a situation in which an event occurred only between the first and last occurrence of another event, during operator seems necessary. '*past#*' is the modified version of '*prev*' which makes its decision based on the desired number of its operand evaluated to true.

For the purpose of increasing the expressiveness of the model, the propositional logic combination of the predicates generated so far was taken into account to create a temporal expression. In this manner, we have a fully functional expressive set of operators that could be used to express different temporal relations between the elements of a history of accesses.

## 4.3 Formal Semantics of Logical Time Authorization Rules

To formalize the semantics of temporal authorization rules, we first define an evaluation function Φ. This function evaluates the predicate *F* of temporal authorization rules at a logical time point *t*, and based on the elements stored in *History Base*. The semantics of such an evaluation is given in first order logic and is reported in Table 2.

**Table 2. Formal semantics of the Φ evaluation function**

$$\Phi_{\mathrm{t}}\big(t_{sh}, done(s,o,a)\big) = \begin{cases} true\ , t \geq t_{sh} \wedge done(t,s,o,a) \in HB \\ false\ , \quad t \leq t_{sh} \vee done(t,s,o,a) \notin HB \end{cases}$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, denied(s,o,a)\big) = \begin{cases} true\ , t \geq t_{sh} \wedge denied(t,s,o,a) \in HB \\ false\ , \quad t \leq t_{sh} \vee denied(t,s,o,a) \notin HB \end{cases}$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, \neg done(s,o,a)\big) = \begin{cases} true\ , \nexists t, t \geq t_{sh}, done(t,s,o,a) \in HB \\ false\ , \exists t, t \geq t_{sh}, done(t,s,o,a) \in HB \end{cases}$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, \neg denied(s,o,a)\big) = \begin{cases} true\ , \nexists t, t \geq t_{sh}, denied(t,s,o,a) \in HB \\ false\ , \exists t, t \geq t_{sh}, denied(t,s,o,a) \in HB \end{cases}$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, prev(A)\big) = \Phi_{\mathrm{t-1}}(t_{sh}, A)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, past\#(A)\big) = \exists t_1, \dots, t_\# \cdot \bigwedge_{k=1}^{\#} \Phi_{\mathrm{t_k}}(t_{sh}, A)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, H(A)\big) = \forall t_i, t_{sh} \leq t_i \leq t, \Phi_{\mathrm{t_i}}(t_{sh}, A)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, sb\#(A_1, A_2)\big) = \exists t_{22}, t_{sh} \leq t_{22} \leq t, \exists t_1, \dots, t_\# \leq t_{22} \cdot \Phi_{\mathrm{t_{22}}}(t_{sh}, A_2) \wedge \bigwedge_{k=1}^{\#} \Phi_{\mathrm{t_k}}(t_{sh}, A_1)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, ab(A_1, A_2)\big) = \big(\exists t_1, t_{sh} \leq t_1 \leq t \cdot \Phi_{\mathrm{t_1}}(t_{sh}, A_1)\big) \rightarrow \big(\exists t_2, t_1 \leq t_2 \leq t \cdot \Phi_{\mathrm{t_2}}(t_{sh}, A_2)\big)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, ss(A_1, A_2)\big) = \forall t_2, t_{sh} \leq t_2 \leq t, \Big(\Phi_{\mathrm{t_2}}(t_{sh}, A_2) \rightarrow \big(\forall t_i, t_2 \leq t_i \leq t \rightarrow \Phi_{\mathrm{t_i}}(t_{sh}, A_1)\big)\Big)$$

$$\Phi_{\mathrm{t}}\big(t_{sh}, during(A_1, A_2)\big) = \Bigg(\Big(\exists t_{min}, t_{sh} \leq t_{min} \leq t \wedge \Phi_{\mathrm{t_{min}}}(t_{sh}, A_2) \wedge \big(\nexists t_x, t_{sh} \leq t_x \leq t_{min} \wedge \Phi_{\mathrm{t_x}}(t_{sh}, A_2)\big)\Big) \wedge \Big(\exists t_{max}, t_{sh} \leq t_{max} \leq t \wedge \Phi_{\mathrm{t_{max}}}(t_{sh}, A_2) \wedge \big(\nexists t_y, t_{max} \leq t_y \leq t \wedge \Phi_{\mathrm{t_y}}(t_{sh}, A_2)\big)\Big)\Bigg) \rightarrow \Big(\forall t_1, t_1 \leq t \rightarrow \big(\Phi_{\mathrm{t_1}}(t_{sh}, A_1) \rightarrow t_{min} \leq t_1 \leq t_{max}\big)\Big)$$

$$\Phi_{\mathrm{t}}(t_{sh}, \neg E) = \Phi_{\mathrm{t}}(t_{sh}, E)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \wedge E_2) = \Phi_{\mathrm{t}}(t_{sh}, E_1) \wedge \Phi_{\mathrm{t}}(t_{sh}, E_2)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \vee E_2) = \Phi_{\mathrm{t}}(t_{sh}, E_1) \vee \Phi_{\mathrm{t}}(t_{sh}, E_2)$$

$$\Phi_{\mathrm{t}}(t_{sh}, \neg E) = \neg\Phi_{\mathrm{t}}(t_{sh}, E)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \wedge E_2) = \Phi_{\mathrm{t}}(t_{sh}, E_1) \wedge \Phi_{\mathrm{t}}(t_{sh}, E_2)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \vee E_2) = \Phi_{\mathrm{t}}(t_{sh}, E_1) \vee \Phi_{\mathrm{t}}(t_{sh}, E_2)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \rightarrow E_2) = \Phi_{\mathrm{t}}(t_{sh}, E_1) \wedge \neg\Phi_{\mathrm{t}}(t_{sh}, E_2)$$

$$\Phi_{\mathrm{t}}(t_{sh}, E_1 \leftrightarrow E_2) = \big(\Phi_{\mathrm{t}}(t_{sh}, E_1) \rightarrow \Phi_{\mathrm{t}}(t_{sh}, E_2)\big) \wedge \big(\Phi_{\mathrm{t}}(t_{sh}, E_2) \rightarrow \Phi_{\mathrm{t}}(t_{sh}, E_1)\big)$$

By the definition of the evaluation function Φ and by the assumption described above, the semantics of authorization rules are presented in Table 3. In the following, $grant\big(t, (s,o,a)\big)$ denotes subject presenting credential *s* is granted to exercise action *a* on object *o*, and analogously, $deny\big(t, (s,o,a)\big)$ denotes the access request of subject having credential *s* for exercising an access *a* on object *o* is denied.

**Table 3. Semantics of logical time authorization rules**

$$\big([t_s, t_f], (s,o,+a), F\big) \Leftrightarrow \forall t \big(t_{sv}, t_{sh} \leq t \leq t_f \wedge \Phi_{\mathrm{t}}(t_{sh}, F)\big) \rightarrow grant\big(t, (s,o,a)\big)$$

$$\big([t_s, t_f], (s,o,-a), F\big) \Leftrightarrow \forall t \big(t_{sv}, t_{sh} \leq t \leq t_f \wedge \Phi_{\mathrm{t}}(t_{sh}, F)\big) \rightarrow deny\big(t, (s,o,a)\big)$$

## 4.4 Access Control

The centric security mechanism in each system is an access control system. Upon receiving an access request in such a system, we need to make a decision whether to grant the requested access or deny it. Following the proposed model of temporal authorization in the previous sections, upon receiving an access request $(s_r, o_r, a_r)$ at time *t*, the access control system performs the following steps:

1. Determine the explicit and implicit valid authorization rules in TAB at time *t* (following the definition of valid authorization rules), satisfying the following conditions:
   - $t_s \leq t \leq min(t_f, t_d)$, $t_d$ is the time of deletion of a specific authorization rule.
   - Temporal predicate *F* is evaluated to true at time *t* (based on $\varphi_t$ evaluation function).
2. Extract the set of valid authorization rules such as $([t_{sv}, t_{sh}, t_f], (s, o, \pm a), F)$ which match the access request. These authorization rules must satisfy, at least, one of the following conditions:
   - $s = s_r$ , $o = o_r$ , $a = a_r$
   - Following the propagation rules of the SBAC model, in the case of a positive action $(+a)$, we have $s_r \prec s$ , $o_r \prec o$ , $a_r \prec a$, and in the case of a negative action $(-a)$, we have $s_r \prec s$ , $o_r \prec o$ , $a \prec a_r$.
3. If there exist just positive valid authorization rule(s) such as $([t_{sv}, t_{sh}, t_f], (s, o, +a), F)$ in MVA, <u>grant</u> the requested access.
4. If there exist just negative valid authorization rule(s) such as $([t_{sv}, t_{sh}, t_f], (s, o, -a), F)$ in MVA, deny the access request,
5. If there exist both positive and negative authorization rules in MVA, resolve the conflict and follow the result.
6. If there exists no valid authorization rule, which matches the requested access, follow the default access policy.
7. Store $done(t, s_r, o_r, a_r)$ in case of the requested access is granted and $denied(t, s_r, o_r, a_r)$ in case of the access request is denied.

## 4.5 Conflict Detection and Resolution

A conflict occurs when two or more access policies cannot be applied in the same time. In access control, due to modal conflict between *matched valid authorizations*, we need a conflict resolution strategy.

### 4.5.1 Conflict Occurrence

In TSBAC, conflict occurs due to the semantic relations between the entities (in the domains of subjects, objects, or actions) and applying the inference rules of SBAC, or due to the sub-interval relationship between temporal authorization rules of TSBAC.

- *Conflict due to the semantic relations between the entities*: as mentioned before, in the domains of subjects and objects, subsumee has all the privileges (positive and negative) of subsumer, but, in the domain of actions, positive access rights is propagated from subsumer to subsumee, while negative access rights is propagated in the opposite direction (that is from subsumee to subsumer). These semantic relationships and propagation of negative and positive authorizations between entities may result in conflicting situations.

### 4.5.2 Conflict Resolution

In access control, due to modal conflicts between valid matched authorization rules (in set MVA), it is required to have a conflict resolution strategy to resolve conflicts. The conflict might be a result of semantic relationships between the entities (i.e. subjects, objects, and actions) and applying the inference rules of SBAC model, or the sub-interval relation between authorizations (i.e. $[t_{sv2}, t_{f2}]$ is a sub-interval of $[t_{sv1}, t_{f1}]$).

TSBAC supports three predefined strategies for conflict resolution; negative authorization rule takes precedence (NTP) strategy, positive authorization rule takes precedence (PTP) strategy, and most specific rule takes precedence. Similar to the default access policy, the conflict resolution strategy is determined by the administrator.

## 5. Architecture

In order to guarantee the applicability of the model and usefulness in semantic based and temporal environments, an architecture for the access control system based on the proposed model is presented.

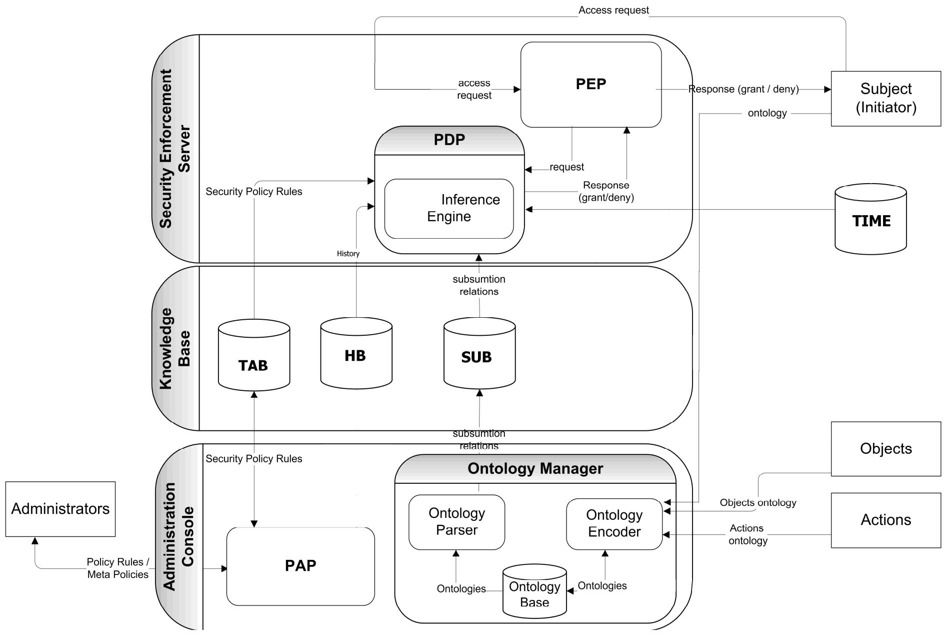

**Figure 1. An architecture for the access control system based on TSBAC model**

## 6. Implementing TSBAC using CLIPS

In order to make inference by CLIPS engine, we must introduce TSBAC to the CLIPS inference engine. This procedure can be summarized into these steps:

- Expressing description logics axioms.
- Expressing subjects ontology, objects ontology, and actions ontology.
- Expressing fix facts of the model.
- Expressing inference rules of SBAC.
- Feeding history base to the inference engine.
- Expressing temporal authorization rules.
- Applying access control and conflict resolution.

The description and details of the implementation could be found in the following URL: http://ce.sharif.edu/~noorollahi/Downloads.html

## 7. Case Study

The required information for applying access control is as follows: two subjects, that is, $s_1$ and $s_2$ exist in the environment, and objects ontology and actions ontology can be seen in Figure 2.

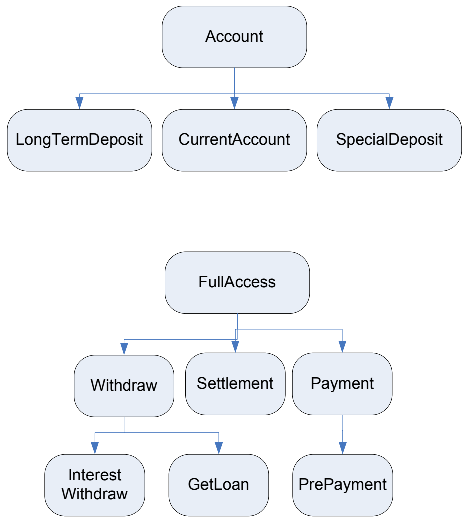

**Figure 2. Objects and actions ontology in the sample banking environment**

Sample temporal authorization rules are as follows:

$$R_1\colon \left( \begin{matrix} [10,\infty], \begin{pmatrix} s_1, LongTermDeposit_1, \\ +InterestWithdraw \end{pmatrix}, \\ H(\neg done(s_1, LongTermDeposite_1, Withdraw)) \end{matrix} \right)$$

$$R_2\colon \left( \begin{matrix} [0,\infty], (s_1, CurrentAccount_1, -Withdraw), \\ sb0(done(all, CurrentAccount_1, Settlement), true) \end{matrix} \right)$$

$$R_3\colon \left( \begin{matrix} [10,\infty], (s_1, CurrentAccount_1, +GetCheque), \\ ss \begin{pmatrix} done(s_1, CurrentAccount_1, GetCheque), \\ \neg denied(all, CurrentAccount_1, pass), 1 \end{pmatrix} \end{matrix} \right)$$

$$R_4: \left( \begin{array}{c} [120, \infty], (s_2, SpecialDeposite_2, +GetLoan), \\ ss \begin{pmatrix} done(s_2, SpecialDeposite_2, PrePayment), \\ done(s_2, SpecialDeposite_2, Payment), 30 \end{pmatrix} \wedge \\ past120\big(done(s_2, SpecialDeposite_2, Payment)\big) \end{array} \right)$$

The elements of history base are as follows:

1: $done(20, s_1, LongTermDeposite_1, Withdraw)$
2: $done(30, s_1, LongTermDeposite_1, Withdraw)$
3: $done(25, s_1, CurrentAccount_1, Settlement)$
4: $done(35, s_1, CurrentAccount_1, Settlement)$
5: $denied(50, s_2, CurrentAccount_1, Pass)$
6: $done(51, s_2, CurrentAccount_1, Pass)$
7: $done(130, s_2, SpescialDeposit_2, PrePayment)$
8: $done\big(140, s_2, SpescialDeposit_2 Payment\big)$
9: $done(150, s_2, SpescialDeposit_2, Payment)$
10: $done(160, s_2, SpescialDeposit_2, Payment)$
11: $done(170, SpescialDeposit_2, Payment)$

- ***First Access Scenario:*** Suppose that request $req_1 = (s_1, LongTermDeposite_1, InterestWithdraw)$ is initiated at the time $t_{req} = 40$. Regarding *R₁* and elements #1 and #2 of History Base, the temporal predicate is evaluated to false, and this access request (*req₁*) is denied (in the case of a closed access policy, or granted in the case of a open access policy). Finally, $denied\begin{pmatrix} 40, s_1, LongTermDeposite_1, \\ InterestWithdraw \end{pmatrix}$ is added to History Base.
- ***Second Access Scenario:*** Suppose that request $req_2 = (s_1, CurrentAccount_1, Withdraw)$ is initiated at the time $t_{req} = 75$. Regarding *R₂* and elements #3 and #4 of History Base, no negative authorization is issued for this request. Therefore, in case of an open access policy, the authorization is granted and $done(75, s_1, CurrentAccount_1, Withdraw)$ is added to History Base, but in case of a negative access policy, the authorization is denied and $denied(75, s_1, CurrentAccount_1, Withdraw)$ is added to History Base.
- ***Third Access Scenario:*** Suppose that request $req_3 = (s_1, CurrentAccount_1, GetCheque)$ is initiated at the time of $t_{req} = 100$. Regarding *R₃* and element #5 that exists in History Base, this access request is denied and $denied(100, s_1, CurrentAccount_1, GetCheque)$ is added to History Base.
- ***Fourth Access Scenario:*** Suppose that request $req_4 = (s_2, SpecialDeposit_2, GetLoan)$ is initiated at the time $t_{req} = 200$. Regarding *R₄* and elements #7…#11 exist in History Base, in spite of existence of element #7 (prepayment) and elements #8 through #11 (monthly settlement), this access request is denied (because the number of monthly payments is less than 120) ,and $denied(200, s_2, SpecialDeposit_2, GetLoan)$ is added to History Base.

## 8. Discussion and Evaluation

The best way to evaluate an access control model is to qualitatively evaluate it against the security requirements of the environment. Moreover, we can take some quantitative criteria into account, but this consideration is only possible if an implementation of an access control system based on the proposed model exists.

### 8.1 Qualitative Evaluation of TSBAC Model

In this section we evaluate TSBAC regarding requirements of semantic-aware environments.

- *Conditional Authorization*: With the existence of temporal operators, TSBAC supports this type of authorization. In this model, due to wide spectrum of temporal operators, and using first order logic operators for combining temporal expressions, conditional authorization is provided, on the basis of existence or non-existence of specific authorizations in the past.
- *Conflict Detection and Resolution*: Conflict occurrence may be a result of semantic relationships between authorizations, or, sub-interval relations between validity constraint intervals. TSBAC detects these conflicts, and resolves them. Different conflict resolution policies include: denials take precedence, positives take precedence, and most specific takes precedence.
- *Supporting History-based Information*: The main feature of TSBAC is that authorizing an access request is done based on granted or denied access requests (done and denied access requests, which are stored in History Base), or, access requests that have not been done or have not been denied in the system (which can be inferred from History Base). These elements could be combined with temporal operators, or first order logics operators to compose temporal expressions.

### 8.2 Quantitative Evaluation of TSBAC

Due to the existence of an implementation for TSBAC, quantitative evaluation of the model is possible. The time and space complexity of the system based on the *logical time* operators of the model is as follows.

- *Time Complexity:* Since every access request is validated at the time of the request, and the process of authorization is based upon searching History Base and evaluating the temporal predicate, due to vast amount of elements of History Base and temporal predicate complexity, access control in TSBAC is time consuming. In order to clarify the subjects mentioned above, we give a brief complexity analysis on *logical time* operators (in case of existence of $n$ elements in History Base) of the proposed model.

$$\boxed{\begin{aligned}&complexity(prev) = n\\&complexity(H) = (t - t_s) \times n\\&complexity(past\#) = \# \times n\\&complexity(sb\#) = (\# + 1)\,n\\&complexity(ab) = n\\&complexity(ss) \leq (t - t_s) \times n\\&complexity(during) = n\end{aligned}}$$

- *Space Complexity:* All of the access requests (granted or denied) are stored in History Base. Storing all the requested accesses in the system, gradually, requires a huge amount of storage space. In case of a vast amount of history elements, and thus, incapability of keeping all these elements on volatile storage, time complexity of access control process is amplified too.

## 9. Conclusion

In this paper, an extension of the Semantic Based Access Control model (SBAC), in order to enhance its capabilities by taking the history of accesses of the system into account was proposed. The proposed model (named TSBAC) uses the same semantic relationships of SBAC, and moreover, it is capable of using temporal relations between the access events occurred in the past (composed as a temporal expression) in specifying authorization rules. In this model, a dynamic aspect are also given to the authorizations by attaching a time interval to each of the authorizations that restricts the validity period of them. These authorization rules (which are composed of base authorization of SBAC, validity time interval, and temporal expression over the history of users' accesses in the system), provides the ability to derive new authorizations based on the occurrence (or not occurrence) of other accesses in the past.

The formal semantics of our authorization rules, plus the access control and conflict resolution procedures were proposed. An architecture for an access control system based on TSBAC was presented. Finally, a banking environment was modeled using TSBAC as a case study.

A generalized history-based access control model that could be applied to other access control policies (such as RBAC) is one of our important future works that will be done.

## References

1. Samarati, P. and S. Vimercati. *Access Control: Policies, models, and mechanisms*. in *Foundations of Security Analysis and Design (FOSAD)*. 2001. Bertinoro, Italy: Springer-Verlag.
2. S. Javanmardi, A. Amini, and R. Jalili. *An Access Control Model for Protecting Semantic Web Resources*. in *Web Policy Workshop*. 2006. Ahens, GA, USA.
3. Javanmardi, S., et al. *SBAC: "A Semantic-Based Access Control Model"*. in *NORDSEC-2006*. 2006.
4. Bonatti, P.A., et al., *Semantic web policies: a discussion of requirements and research issues.* ESWC, 2006. **2006**: p. 712-724.
5. Rabitti, F., et al., *A Model of Authorization for Next-Generation Database Systems.* ACM Transactions on Database Systems, 2001. **16**(1): p. 88-131.
6. Qin, L. and V. Atluri. *Concept-level access control for the Semantic Web*. in *2003 ACM workshop on XML security*. 2003.
7. Brewer, D.F.C. and M.J. Nash. *The Chinese Wall Security Policy*. in *IEEE Symposium on Security and Privacy*. 1989. Oakland, California.
8. Dias;, P., C. Ribeiro;, and P. Ferreira, *Enforcing History-Based Security Policies in Mobile Agent Systems*. 2003.
9. Edjlali, G., A. Acharya, and V. Chaudhary. *History-based access control for mobile code*. in *5th ACM conference on Computer and communications security*. 1998.
10. Abadi, M. and C. Fournet. *Access control based on execution history*. in *10th Annual Network and Distributed System Security Symposium*. 2003.
11. Bertino, E., et al., *A temporal access control mechanism for Database Systems.* IEEE Trans. Knowl. Data Eng, 1996. **8**(1): p. 67-80.
12. Thomas, R.K. and R.S. Sandhu. *Sixteenth National Computer Security Conference*. 1993. Baltimore, Md.
13. Bertino, E., C. Bettini, and P. Samarati. *A temporal authorization model*. in *Second ACM Conference on Computer and Communications Security*. 1994. Fairfax, Va.
14. Bertino, E., et al., *An access control model supporting periodicity constraints and temporal reasoning*. ACM Trans. Database Systems, 1998. **23**(3): p. 231-285.
15. Ruan, C., *Decentralized Temporal Authorization Administration*. 2003.
16. Bertino, E., et al., *Temporal authorization bases: From specification to integration.* Journal of Computer Security, 2000. **8**: p. 309-353.
17. Emerson, E.A., *Temporal and modal logic.* Handbook of Theoretical Computer Science, 1990. **B: Formal Models and Sematics (B)**: p. 997-1072.