Proceedings of the International Multiconference on
Computer Science and Information Technology, pp. 827 – 836

ISBN 978-83-60810-14-9
ISSN 1896-7094

A Semantic Aware Access Control Model with Real Time Constraints on History of Accesses

Ali Noorollahi Ravari
Network Security Center, Computer Engineering Department,
Sharif University Of Technology, Tehran, Iran
Email: noorollahi@ce.sharif.edu

Morteza Amini, Rasool Jalili
Network Security Center, Computer Engineering Department,
Sharif University Of Technology, Tehran, Iran
Email: { m_amini@ce., 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. SBAC model considers this issue in its decision making process. However, time plays a crucial role in new computing environments which is not supported in the model. In this paper we introduce the Temporal Semantic Based Access Control model (TSBAC), 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. 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 the proposed model is presented, and finally, we discuss and evaluate TSBAC.**

I. Introduction

ACCESS control is a mechanism that allows owners of resources to define, manage and enforce access conditions applicable to each resource [1]. 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 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 [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 previously, that is, we use SBAC (as the base model), and associate a temporal expression with each authorization. Due to the nature of some application domains (such as the banking environment), a real representation of time is required to be used for modeling temporal dependencies between history of accesses. So, in this paper, we use real time operators to impose constraints on elements of History Base. Furthermore, a temporal interval bounds the scope of the temporal expressions (e.g., [1,20] shows that the authorization is valid for time interval starting at '1' and ending at '20'). Thus, the main feature provided by TSBAC is the possibility of specifying authorization rules which express temporal dependencies among authorizations. These rules allow derivation of new authorizations based on the presence or absence of other authorizations in specific past time instants (stored in *History Base* in the form of `done(t,s,o,a)` and `denied(t,s,o,a)`)

A formal semantics is defined for temporal authorizations. The subject of Temporal Authorization Base (TAB) administration and conflicting situations are investigated and resolved. An architecture for the access control system based on TSBAC, and an evaluation is presented.

The rest of this paper is organized as follows: in Section 2 we discuss the related works on 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 and offer the formal semantics followed by a brief description of administration of the authorization base 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 we give a brief evaluation of our work. Finally, section 7 concludes the paper.

II. Related Work

Access control systems for protecting Web resources along with credential based approaches for authenticating users have been studied in recent years [1]. With the advent of Semantic Web, new security challenges were imposed to security systems. Bonatti *et al.*, in[4] have discussed open issues in the area of policy 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 in [5]. A concept level access

control model which considers some semantic relationships in the level of concepts in the object domain is proposed in [6]. The main work on SBAC, which is the basis of our model, is proposed in [2, 3] by Javanmardi *et al.*. SBAC 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 past 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 which considers the history of control transfers, rather than a history of sensitive requests, is presented in [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*. in [11], an authorization is specified as $(time,auth)$, where $time=(t_b,t_e)$ as the time interval, and $auth=(s,o,m,pn,g)$ as an authorization. Here, t_b and t_e represent the start and end times respectively, during which *auth* is valid; *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.* in [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.

III. Preliminaries

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

A. 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 Σ, $C<D$ denotes that *D* subsumes *C* in Σ. This reasoning based on subsumption proves that *D* (the subsumer) is more general than *C* (the subsumee).

By reducing all semantic relationships to the subsumption, the following propagation rules are enough:

- *Propagation in subjects domain*: Given $(si,o,\pm a)$, if $sj<si$ then $(sj,o,\pm a)$.
- *Propagation in objects domain*: Given $(s,oi,\pm a)$, if $oj<oi$ then $(s,oj,\pm a)$.
- *Propagation in actions domain*:
  - Given $(s,o,+ai)$, if $aj<ai$ then $(s,o,+aj)$.
  - Given $(s,o,-aj)$, if $aj<ai$ then $(s,o,-ai)$.

B. Modeling of Time

In this paper, we assume a real representation of time. It is worthwhile to note that, we suppose that the response time of the access control system is trivial and thus we ignore the time duration required by the system to check whether a requested access is granted or denied. This assumption allows us to take an access request time as the access time recorded in the history.

A good representation of time for instantaneous events, if possible, is using an absolute dating system. This involves time stamping each event with an absolute real-time. For instance, a convenient dating scheme could be a tuple consisting of the *year*, *month in the year*, *day in month*, *hour in the day*, *minutes*, and *seconds*. For example, $(2008\ 1\ 20\ 10\ 4\ 50)$ would be the 20th day of January 2008, at 10:04 (AM) and 50 seconds. The big advantage of dating schemes is that they provide for constant time algorithms for comparing times and use only linear space in the number of items represented.

Time comparisons are reduced to simple numeric comparisons. Date-based representations are only usable, however, in applications where such information is always known, i.e. applications where every event entered has its absolute date identified. There are many applications where this is a reasonable assumption; for instance, databases of transactions on a single machine, say a central machine maintaining banking records. In addition, with absolute dating, we also

have information about the duration of time between events (we simply subtract the date of the later event from the date of the earlier one).

IV. Temporal Semantic based Access Control Model

In this section we introduce our authorization model, Temporal Semantic based Access Control model (TSBAC), which is an extension of the SBAC model. In TSBAC, we extend the basic authorization model in two directions: adding authorization validation time interval, and associating a temporal expression over a *History Base* (history of users' accesses).

A. Temporal Authorization Rules with Real Time Scheme

In TSBAC we consider a temporal constraint to be associated with each authorization. This constraint is based on the privileges granted to subjects of the system (on objects), or access requests denied, in a specific real *time point* in the past. These elements of history are stored in *History Base*, in the form of $\text{done}_{t,s,o,a}$ and $\text{denied}_{t,s,o,a}$. We refer to an authorization, together with a temporal constraint and a validation time interval, as a temporal authorization rule. A temporal authorization rule is defined as follows.

Definition (Temporal Authorization Rule): A temporal authorization rule is a triple $([t_s,t_f],(s,o,\pm a),F)$, where $t_s \in$ real-time-sceheme, $t_f \in$ real-time-scheme, and $t_s \leq t_f$. In this notation, $[t_s,t_f]$ represents the authorization validation time interval, and formula F is a temporal constraint which is formally defined as in Table 1.

TABLE 1.
DEFINITION OF TEMPORAL PREDICATE F

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

Temporal authorization rule $([t_s,t_f],(s,o,\pm a),F)$ states that subject s is allowed (or not allowed) to exercise access a on object o in the interval $[t_s,t_f]$, including time instants t_s and t_f, in the case that F is evaluated to true.

Definition (Temporal Authorization Base): A temporal authorization base (TAB) is a set of temporal authorization rules in the form of $([t_s,t_f],(s,o,\pm a),F)$, where $t_s \in$ real-time and $t_f \in$ real-time.

Definition (History Base): A History Base is a set of authorizations and time points, in the form of $\text{done}_{t,s,o,a}$ which means access a has been granted to subject s on object o at real time point t, and $\text{denied}_{t,s,o,a}$ which means the system has denied access a on object o at real time point t requested by subject s.

B. Informal Meaning of Temporal Authorization Rules

The intuitive meaning of temporal authorization rules is as follows. In these statements *auth* is representative of $(s,o,\pm a)$.

- $([t_s,t_f],auth,done(s,o,a))$: Authorization *auth* is valid in all time instants t, in interval $[t_s,t_f]$, in which $done(s,o,a)$ is evaluated to true. In other words, *auth* is valid at time t, if $done(t,s,o,a)$ exists in HB.
- $([t_s,t_f],auth,denied(s,o,a))$: Authorization *auth* is valid in all time instants t, in interval $[t_s,t_f]$, in which $denied(s,o,a)$ is evaluated to true. In other words, *auth* is valid at time t, if $denied(t,s,o,a)$ exists in HB.
- $([t_s,t_f],auth,\sim done(s,o,a))$: Authorization *auth* is valid in all time instants t, in interval $[t_s,t_f]$, in which $done(s,o,a)$ is not evaluated to true.
- $([t_s,t_f],auth,\sim denied(s,o,a))$: Authorization *auth* is valid in all time instants t, in interval $[t_s,t_f]$, in which $denied(s,o,a)$ is not evaluated to true.
- $([t_s,t_f],auth,prev(A))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$, if A is evaluated to true at the previous moment (t-1). The previous *time point* is determined due to the precision of selected time scheme. For example, if the precision of time is "seconds", the tuple to represent time is of the form of (yyyy dd hh mm ss), so the previous time point is (yyyy dd hh mm (ss-1)). In short, to calculate the previous time point, we simply subtract the numerical representation of time by one. So, PrvTimePoint(yyyy dd hh mm ss)=(yyyy dd hh mm ss)-1 Figure 1 gives a more comprehensible view of the operation of this operator.

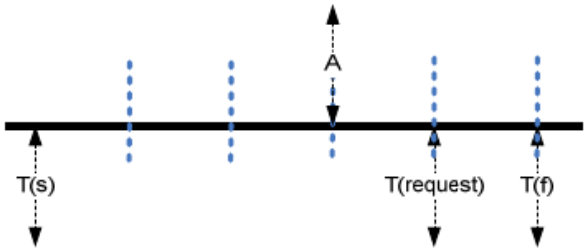

Figure 1. Operation of the *prev* operator on real time axis.

■

- $([t_s,t_f],auth,past\#(A))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$ if F is evaluated to true for # of times from t_s till t. 2 gives a more comprehensible view of the operation of this operator.

Example 1. Subject s_1 is allowed to get another loan (on $Deposit_1$), if he has paid all his (past 36) payments. It is assumed that the date of getting the first loan is 2004/07/01. This rule can be expressed as:

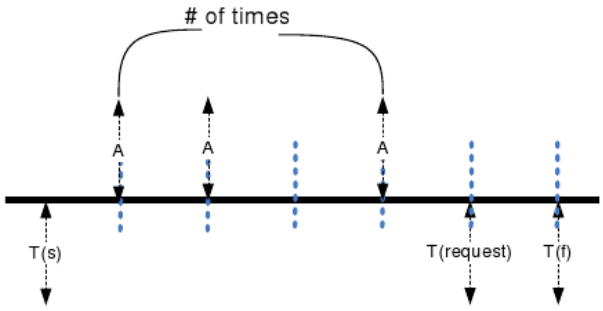

Figure 2. Operation of *past#* operator on real time axis.

$$R: \begin{pmatrix} [2004\ 7\ 1, \infty], (s_1, Deposit_1, +getLoan), \\ past36\big(done(s_1, Deposit_1, payment)\big) \end{pmatrix}.$$

■

- $([t_s,t_f],auth,H(A,chunk))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$, if F is evaluated to true, at least once in each time interval of length *chunk*, from t_s till t. *chunk* is used to reduce the precision of the operator and relax its operation. Figure 3 gives a more comprehensible view of this operator operation.

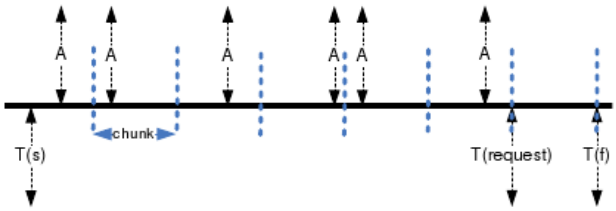

Figure 3. Operation of *H* operator on real time axis.

Example 2. Due to insurance rules, everybody can be insured, if he pays for it monthly. This rule for subject s_1 that has entered the system since January, 2005 is expressed as: (note that it is assumed a month to be 30 days)

$$R: \begin{pmatrix} [2005\ 1, \infty], (s_1, specialIns, +takeAdvantage), \\ H(done(s_1, insDeposit_1, settlement), 30) \end{pmatrix}.$$

■

All of the operators studied so far, has only one element as their argument. It means that they make their decision (to grant or deny a request) based on presence or absence of just one element in the history base (or first order logic combination of them, but not the temporal relation between two or more of them). In some applications, we need to decide based on the relation between elements of HB. So, TSBAC uses operators that consider the temporal relation between two elements of the history base.

- $([t_s,t_f],auth,sb\#(A_1,A_2))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$ if A_1 is evaluated to true # of times before the last occurrence of A_2, from t_s till t. 4 gives a more comprehensible view of the operation of this operator.

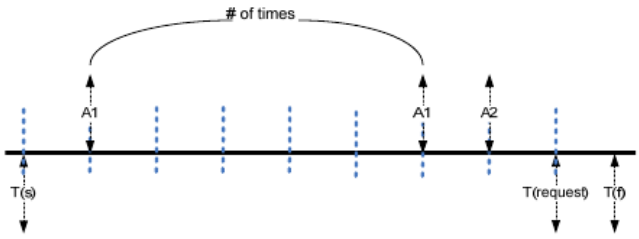

Figure 4. Operation of the *sb#* operator on real time axis

■

- $([t_s,t_f],auth,ab(A_1,A_2))$: Authorization *auth* is valid at the time of request (t) in the interval $[t_s,t_f]$ if, if A_1 is evaluated to true in $t't_s \le t' \le t$, then there exist a time point t'' $t' \le t'' \le t$, in which A_2 is evaluated to true. In the other words, A_1 is evaluated to true in time instants before the evaluation of A_2 to true, from t_s till the time of request (t). Figure 6 gives a more comprehensible view of the operation of this operator.

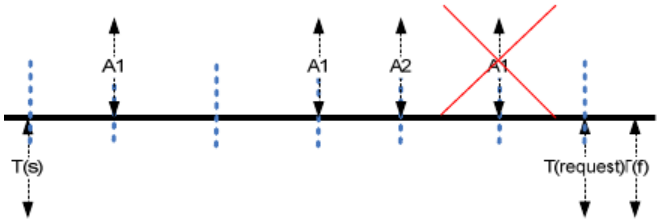

Figure 5. Operation of the *ab* operator on real time axis

Example 3. Subject s_1 is allowed to get loan on his account ($Account_1$), if he has not withdrawn money from his account since applying for it (that is 2007/01/20). This rule can be expressed as:

$$\begin{pmatrix} [2007\ 20, \infty], (s_1 Account_1, +getLoan), \\ ab\begin{pmatrix} \sim done(s_1 Account_1, withdraw), \\ done(s_1 Account_1, applyForLoan) \end{pmatrix} \end{pmatrix}.$$

■

- $([t_s,t_f],auth,ss(A_1,A_2,chunk))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$, if A_1 is evaluated to true, at least one time in all time intervals of length chunk, from the first occurrence of A_2 in interval $[t_s,t]$. Figure 6 gives a more comprehensible view of the operation of this operator.

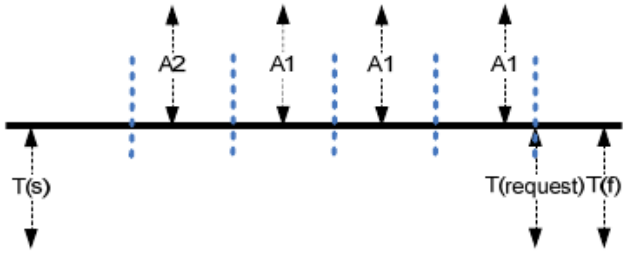

Figure 6. Operation of *ss* operator on real time axis

Example 4. Subject s_1 is on the car waiting list, if he paid a prepayment (2006/02/01), and since then he has been paying a defined payment monthly. This rule can be expressed as:

$$\begin{pmatrix} [2006\,31, \infty], (s_1, carWaitingList, +get), \\ ss\begin{pmatrix} done(s_1, Account_1, payment), \\ done(s_1, Account_1, prePayment), 30 \end{pmatrix} \end{pmatrix}.$$

■

- $([t_s,t_f],auth,during(A_1,A_2))$: Authorization *auth* is valid at the time of request (t) in interval $[t_s,t_f]$ if A_1 is not true before the first, or after the last time instant in which A_2 is true. Figure 7 gives a more comprehensible view of the operation of this operator.

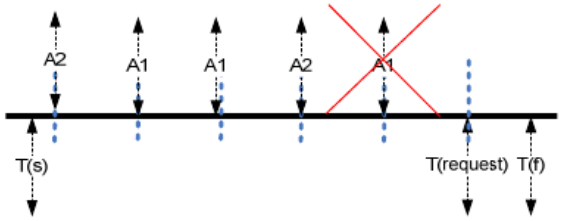

Figure 7. Operation of *during* operator on real time axis

■

- $([t_s,t_f],auth,\sim E)$: Authorization *auth* is valid for each time instant t in interval $[t_s,t_f]$ in which E is *not* evaluated to true.
- $([t_s,t_f],auth,E_1 \wedge E_2)$: Authorization *auth* is valid for each time instant t in the interval $[t_s,t_f]$ in which E_1 and E_2 are both evaluated to true.
- $([t_s,t_f],auth,E_1 \vee E_2)$: Authorization *auth* is valid for each time instant t in the interval $[t_s,t_f]$ in which E_1 or E_2 or both of them are evaluated to true.
- $([t_s,t_f],auth,E_1 \rightarrow E_2)$: Authorization *auth* is valid for each time instant t in the interval $[t_s,t_f]$ in which, if A_1 is evaluated to true, then A_2 is also evaluated to true.
- $([t_s,t_f],auth,E_1 \leftrightarrow E_2)$: Authorization *auth* is valid for each time instant t in the interval $[t_s,t_f]$, in which A_1 is evaluated to true if and only if A_2 is evaluated to true.

C. Formal Semantics of Temporal Authorization Rules

Next we formalize the semantics of authorization rules described so far.

Definition (Valid Authorization): an authorization $(s,o,\pm a)$ is valid at time t, if one of the following situations occurred:

1. At time t, a temporal authorization rule $([t_s,t_f],(s,o,\pm a),F)$ with $t_s \le t \le t_f$ exists in TAB and F is evaluated to true based on the elements exist in *History Base* (we define function f for performing such an evaluation),
2. There exists a temporal authorization rule $([t_s,t_f],(s',o',\pm a'),F)$ in TAB with $t_s \le t \le t_f$ in which F is evaluated to true, and $(s',o',\pm a')$ is derived from $(s,o,\pm a)$ following the inference rules of SBAC.

- To formalize the semantics of temporal authorization rules, we first define an evaluation function f_{real}. This function evaluates the predicate F of temporal authorization rules at a real 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. The semantics of a set X of temporal authorization rules, denoted by $S(X)$, is the conjunction of the first order formulas corresponding to each element in the set.
- Note that a temporal authorization rule can be removed and therefore not be applicable anymore for the derivation of authorizations. In the formalization we take this possibility into account, by associating with each temporal authorization rule the time t_d at which it is removed. Note that time t_d is not a constant and it is not known from the former. We use it as shorthand for expressing the point up to which a temporal authorization rule is applicable. A function *removed()* can be defined, which, given a temporal authorization rule, X, and a time t returns *false* if at time t, X is still present in the TAB, and ,*true*, otherwise. Time t_d is the smallest time t for which function *removed(t , X)* returns *true*.

TABLE 2.
FORMAL SEMANTICS OF THE F_{REAL} EVALUATION FUNCTION

$$f_{real}\left(t,t_s,t_f,done(s,o,a)\right) = \begin{cases} true, t \in [t_s,t_f] \wedge done(t,s,o,a) \in HB \\ false, t \notin [t_s,t_f] \vee done(t,s,o,a) \notin HB \end{cases}$$

$$f_{real}\left(t,t_s,t_f,denied(s,o,a)\right) = \begin{cases} true, t \in [t_s,t_f] \wedge denied(t,s,o,a) \in HB \\ false, t \notin [t_s,t_f] \vee denied(t,s,o,a) \notin HB \end{cases}$$

$$f_{real}\left(t,t_s,t_f,\sim done(s,o,a)\right) = \begin{cases} true, \nexists t, t \in [t_s,t_f] \cdot done(t,s,o,a) \in HB \\ false, \exists t, t \in [t_s,t_f] \cdot done(t,s,o,a) \in HB \end{cases}$$

$$f_{real}\left(t,t_s,t_f,\sim denied(s,o,a)\right) = \begin{cases} true, \nexists t, t \in [t_s,t_f] \cdot denied(t,s,o,a) \in HB \\ false, \exists t, t \in [t_s,t_f] \cdot denied(t,s,o,a) \in HB \end{cases}$$

$$f_{real}\left(t,t_s,t_f,prev(A)\right) = f\left(t-1,t_s,t_f,A\right)$$

$$f_{real}\left(t,t_s,t_f,past\#(A)\right) = \exists t_1,\dots,t_{\#} \cdot \bigwedge_{k=1}^{\#} f\left(t_k,t_s,t_f,A\right)$$

$$f_{real}\left(t,t_s,t_f,H(A,chunk)\right) = \exists t_0,\dots,t_{\lfloor (t-t_s)/chunk \rfloor - 1} \le t \cdot \bigwedge_{k=0}^{\lfloor (t-t_s)/chunk \rfloor - 1} f(t_k, t_s + i \times chunk, t_s + (i+1) \times chunk, A)$$

$$f_{real}\left(t,t_s,t_f,sb\#(A_1,A_2)\right) = \exists t_{22} \le t, \exists t_1,\dots,t_{\#} \le t_{22} \cdot f\left(t_{22},t_s,t_f,A_2\right) \wedge \bigwedge_{k=1}^{\#} f\left(t_k,t_s,t_f,A_1\right)$$

$$f_{real}\left(t,t_s,t_f,ab(A_1,A_2)\right) = \left(\exists t_1, t_1 \le t \cdot f\left(t_1,t_s,t_f,A_1\right)\right) \rightarrow \left(\exists t_2, t_1 \le t_2 \le t \cdot f\left(t_2,t_s,t_f,A_2\right)\right)$$

$$f_{real}\left(t,t_s,t_f,ss(A_1,A_2,chunk)\right) = \exists t_0,\ldots,t_{\lfloor (t-t')/chunk \rfloor - 1} \le t \cdot f\left(t',t_s,t_f,A_2\right) \wedge \bigwedge_{k=0}^{\lfloor (t-t')/chunk \rfloor - 1} f(t_k, t' + i \times chunk, t' + (i+1) \times chunk, A_1)$$

$$f_{real}\left(t,t_s,t_f,during(A_1,A_2)\right) = \left(\left(\exists t_{min}, t_{min} \le t \wedge f\left(t_{min},t_s,t_f,A_2\right) \wedge \left(\nexists t_x, t_x \le t_{min} \wedge f\left(t_x,t_s,t_f,A_2\right)\right)\right) \wedge \left(\exists t_{max}, t_{max} \le t \wedge f\left(t_{max},t_s,t_f,A_2\right) \wedge \left(\nexists t_y, t_{max} \le t_y \wedge f\left(t_y,t_s,t_f,A_2\right)\right)\right)\right) \rightarrow \left(\forall t_1, t_1 \le t \rightarrow \left(f\left(t_1,t_s,t_f,A_1\right) \rightarrow t_{min} \le t_1 \le t_{max}\right)\right)$$

$$f_{real}\left(t,t_s,t_f,\sim E\right) = \sim f\left(t,t_s,t_f,E\right)$$

$$f_{real}\left(t,t_s,t_f,E_1 \wedge E_2\right) = f\left(t,t_s,t_f,E_1\right) \wedge f\left(t,t_s,t_f,E_2\right)$$

$$f_{real}\left(t,t_s,t_f,E_1 \vee E_2\right) = f\left(t,t_s,t_f,E_1\right) \vee f\left(t,t_s,t_f,E_2\right)$$

$$f_{real}\left(t,t_s,t_f,E_1 \rightarrow E_2\right) = f\left(t,t_s,t_f,E_1\right) \wedge \sim f\left(t,t_s,t_f,E_2\right)$$

$$f_{real}\left(t,t_s,t_f,E_1 \leftrightarrow E_2\right) = \left(f\left(t,t_s,t_f,E_1\right) \rightarrow f\left(t,t_s,t_f,E_2\right)\right) \wedge \left(f\left(t,t_s,t_f,E_2\right) \rightarrow f\left(t,t_s,t_f,E_1\right)\right)$$

By the definition of evaluation function freal and by the assumption described above, the semantics of authorization rules are in Table 3 . In the following, grantt,s,o,a denotes subject s is granted to exercise action a on object o and analogously denyt,s,o,a denotes the access request of s for exercising an access a on object o is denied.

TABLE 3.
SEMANTICS OF REAL TIME AUTHORIZATION RULES

$$\left([t_s,t_f],(s,o,+a),F\right) \Leftrightarrow \forall t \left(t_s \le t \le min\left(t_f,t_d-1\right) \wedge f_{real}\left(t,t_s,t_f,F\right)\right) \rightarrow grant\left(t,(s,o,a)\right)$$

$$\left([t_s,t_f],(s,o,-a),F\right) \Leftrightarrow \forall t \left(t_s \le t \le min\left(t_f,t_d-1\right) \wedge f_{real}\left(t,t_s,t_f,F\right)\right) \rightarrow deny\left(t,(s,o,a)\right)$$

D. Access Control

The centric security mechanism in each system is an access control system. By 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 $sr,or.ar$ 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:
   - $t_s \le t \le min(t_f,t_d)$
   - Temporal predicate *F* is evaluated to true at time *t* (based f_{real} evaluation function).
2. Extract the set of valid authorization rules such as $([t_s,t_f],(s,o,\pm a),F)$ 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 $sr<s$, $or<o$, $ar<a$, and in the case of a negative action ($-a$), we have $sr<s$, $or<o$, $a<ar$.
3. If there exist just positive valid authorization rule(s) such as $([t_s,t_f],(s,o,+a),F)$ in MVA, grant the requested access,
4. If there exist just negative valid authorization rule(s) such as $([t_s,t_f],(s,o,-a),F)$ in MVA, deny the access request,
5. If there exist both positive and negative authorization rules in MVA, do conflict resolution and follow the result,
6. If there exists no valid authorization rule, which matches the requested access, follow the default access policy,
7. Record $done(s_r,o_r.a_r)$ in the case that the requested access is granted, and $denied(s_r,o_r,a_r)$ in the case that the access request is denied.

In this model, the default access policy might be *positive (open)* to grant all undetermined accesses, or *negative (close)* to deny them. The default access policy is determined by the administrator.

E. 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.

1) Conflict Occurrence

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

- *Conflict due to semantic relations between the entities*: as mentioned before, in the domains of subjects and objects, the subsumee has all the privileges (positive and negative) of the subsumer, but, in the domain of actions, positive access rights is propagated from sub-

sumer to subsumee, while negative access rights is propagated in the opposite direction (that is from subsumee to subsumer). These semantic relationships and the propagation of negative and positive authorizations between the entities may result in conflicting situations. As an example, consider the following History Base, semantic relation, and authorization rules:

$$\textbf{HB Contains}:$$
$$done(10, Student, doc_1, read),$$
$$done(8, Ali, doc_1, write)$$
$$R_1: \begin{pmatrix} [0,25], (Student, doc_1, +read), \\ prev(done(Student, doc_1, read)) \end{pmatrix}$$
$$R_2: \begin{pmatrix} [0,25], (Ali, doc_1, -read), \\ prev(done(Student, doc_1, read)) \end{pmatrix}$$
$$\textbf{Subjects Ontology}:$$
$$Ali \prec Student$$

If *Ali* requests a *read* access at time *11* , due to R_2 authorization rule, this access is denied, but due to the R_1 authorization rule we have `Student,doc1,+read` , and based on the sample subjects ontology, the *read* access is also granted for *Ali* . So, in this situation we are confronted with a conflicting situation, due to the semantic relationships between the entities.

- *Conflict due to sub-interval relationship between authorization rules*: as an example, consider the following HB, and authorization rules:

$$\textbf{HB Contains}:$$
$$done(8, Ali, doc_1, read), done(9, Ali, doc_1, read),$$
$$done(10, Ali, doc_1, read)$$
$$R_1: \begin{pmatrix} [8,20], (Ali, doc_1, +read), \\ H(done(Ali, doc_1, read)) \end{pmatrix}$$
$$R_2: \begin{pmatrix} [0,20], (Ali, doc_1, -read), \\ prev(done(Ali, doc_1, read)) \end{pmatrix}$$

If *Ali* requests a *read* access at time *11* , due to R_2 authorization rule, this access is denied, but is granted due to the R_1 authorization rule. So, we are confronted with a conflicting situation based on the sub-interval relationship between R_1 and R_2.

2) Conflict Resolution

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

F. Temporal Authorization Base Administration

Authorization rules can be changed upon the execution of administrative operations. In this paper, we consider a centralized policy for administration of authorizations where administrative operations can be executed only by the administrator.

Administrative operations allow the administrator to add, remove, or modify (a *remove* operation followed by an *add* operation) temporal authorizations rules. Each temporal authorization rule in the TAB is identified by a unique label assigned by the system at the time of its insertion. The label allows the administrator to refer to a specific temporal authorization rule upon execution of administrative operations. A brief description of the administrative operations is as follows:

addRule: To add a new temporal authorization rule. When a new rule is inserted, a label (*rule identifier* or *rid*) is assigned by the system.

dropRule: To drop an existing temporal authorization rule. The operation requires as argument, the label of the rule to be removed.

V. Architecture

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

Several frameworks has been proposed for access control and security in recent years, which, the standard framework of the ITU-T [17] for access control and the standard framework OASIS under the name XACML [18] are the most popular ones. during the design of our model, we have tried to include the elements of both the frameworks mentioned above, especially, the XACML. The major elements of the system are illustrated in .8 The system is composed of a number of *internal* and *external* elements which are described next.

A. External Entities

The major external entities interacting the system are:

Subject: A subject can be a person, a service, or a machine that tries to access resources or objects in a semantic based environment.

Environment: The set of *attributes* that are relevant to an *authorization decision* and are independent of a particular *subject, resource*, or *action*.

Objects and access rights: entities which provide ontologies in the domains of subject, object, and action. These ontologies and the semantic relations between them are helpful in propagation and inference of new security rules.

B. Administration Console

This console enables the security administrator to describe meta-policies and also description and administration of the ontologies. The major components of the administration console are as follows:

Policy Administration Point (PAP): The system entity that creates a policy or policy set.

Ontology manager: Gathers and updates ontologies in domains of subjects, objects, and actions and also reduces the semantic relations to the subsumption relation.

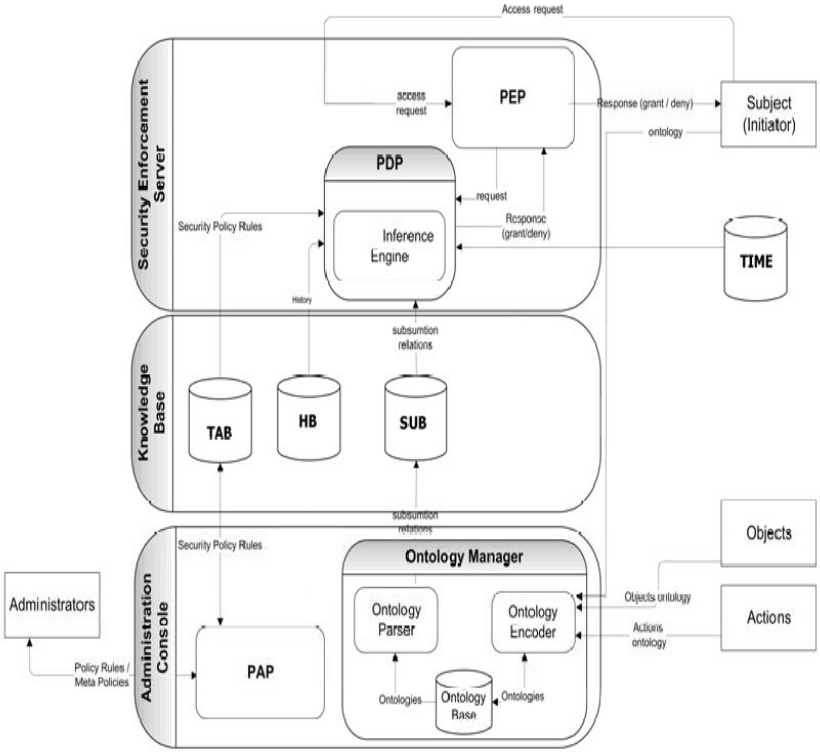

Figure 8. An architecture for an access control system based on TSBAC model

C. Knowledge Base

As described in the model, the knowledge base is composed of the set of *authorization rules* in TAB, *history of authorizations* (HB), *subsumption relations* between concepts (SUB), and a *time counter* (TIME). Inference of implicit authorization rules is based upon the facts and rules in the knowledge base.

D. Security Enforcement Server

This element manages inference of the implicit authorization rules and applying them in access control requests. Major sub-elements of this element are as follows:

Policy Decision Point (PDP): The system entity that evaluates *applicable policy* and renders an *authorization decision* by making use of an inference engine, based on facts and rules in the knowledge base.

Policy Enforcement Point (PEP): The system entity that performs access control, by making decision requests and enforcing authorization decisions.

VI. Discussion and Evaluation

Evaluating authorization models and access control mechanisms, and presenting acceptable criteria in this domain, has been a problem in security and access control zone. Comparing security models with each other, due to differences between them in security definition, seems improper. Security is a comparative quality, and assumptions in security definitions in an environment and security requirements in that environment makes distinctive differences in designing the model. The best way to evaluate a model is to qualitatively scrutiny the model to ensure accordance with security requirements of environment under custody. Moreover, we can take some quantitative criteria into account, but this consideration is possible if an implementation exists for the access control system based on the model.

A. Qualitative Evaluation of TSBAC Model

In this section, we evaluate TSBAC, regarding requirements of semantic and history based environments.

- Fine-grained and Coarse-grained Authorization: TSBAC allows definition of policies for entities in three domains of access control (namely subjects, objects, and actions), so it provides coarse-grained authorization. Moreover, with the existence of ontology, and possibility of defining entities to the individual level, fine-grained authorization is provided.
- 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.
- Different Policies and Expressing Exceptions: TSBAC provides synthetic policy (including negative and positive authorizations). Moreover, by using ontology in domains of subjects, objects, and actions, and utilizing different authorization propagation methods, expressing exceptions and synthetic policies is possible.

- 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, most specific takes precedence, and newer overrides older.
- Ease of Implementation and Integration with Semantic Web technologies: Security models designed for Semantic Web should be compatible with the technology infrastructure under it. In other words, the implementation of security mechanisms should be possible based on the semantic expression models. SBAC is designed based on the widely accepted semantic web languages, OWL and SWRL; therefore its implementation can be easily achieved by existing tools designed for working with these languages.
- 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.
- Interoperability : Interoperating across administrative boundaries is achieved through exchanging authorizations for distributing and assembling authorization rules. The ontological modeling of
- Authorization rules in SBAC results in a higher degree of interoperability compared with other approaches to access control. This is because of the nature of ontologies in providing semantic interoperability.
- Generality: Modeling different domains of access control has added a considerable generality to the model. In the subject domain, TSBAC uses credentials which are going to be universally used for user authentication. In the domain of object, different kinds of resources such as web pages or web services can be modeled and can be identified by their URI in authorization rules.

B. *Quantitative Evaluation of TSBAC*

- 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 some situations, in order to evaluate the temporal predicate, we need to scrutinize the existence of "not-done" or "not denied" requests, and this adds to the time complexity of access control process.

In order to clarify the subjects mentioned above, we give a brief complexity analysis on real time operators (in case of existence of *n* elements in History Base) of the model:

complexityprev=n
complexityH=t-ts/chunk×n
complexitypast#=#×n
complexitysb#=#+1n
complexityab=n
complexityss≤t-ts/chunk×n
complexityduring=n

- 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.

VII. Conclusions and Future Work

Access control and its requirements in new computing environments, semantic aware access control, and history based access control have been discussed in this paper. Based on the Semantic Based Access Control model (SBAC), and to enhance the capabilities of the model, a semantic aware access control model, which takes the history of accesses of the system into account (TSBAC) is proposed. TSBAC uses the same semantic relationships of the SBAC model, and moreover, it is capable of using temporal relations between authorizations in applying access control. Specifically, TSBAC assigns a temporal expression (over users' history of accesses) to each authorization that expresses the conditions under which the authorization applies. A constraining time interval restricts the interval of validity of the authorization. These authorization rules (which are composed of base authorization of SBAC, constraining time interval, and temporal expression), provides the ability to derive new authorizations based on existence (or non-existence) of other authorizations in the past.

We also proposed formal semantics of our authorization rules. Access control, and conflict detection and resolution presented. An architecture for the access control system based on TSBAC was presented.

Producing preconditions of applying temporal logics operators in Java language, and using these preconditions in CLIPS inference engine in order to apply access control can be considered as some future works.

One of the main deficiencies of TSBAC is the lack of a formal proof for soundness and completeness of temporal operators of the model. On the other hand, a generalized history-based access control model that could be applied to other access control policies (such as RBAC) is one of the important works that could be done.

References

[1]. Samarati, P. and S.C.d. Vimercati, *Access control: Policies, models, and mechanisms.* Foundations of Security Analysis and Design, LNCS, 2001. **2171**: p. 137-196.

[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 TODS, 1991. **16**(1): p. 87-99.

[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]. ISO/IEC:10181-3. *Information Technology - Open Systems Interconnection - Security Frameworks for Open Systems: Access Control Framework*. 1995.

[18]. Moses, T., *eXtensible Access Control Markup Language, Version 2.0*. 2005.