Context-Aware Provisional Access Control

Amir Reza Masoumzadeh, Morteza Amini, and Rasool Jalili

Computer Engineering Department
Sharif University of Technology
Tehran, Iran
{masoumzadeh@ce.,m_amini@ce.,jalili@}sharif.edu

Abstract. High heterogeneity and dynamicity of pervasive computing environments introduces requirement of more flexible and functional access control policies. The notion of provisional actions has been defined previously to overcome the insufficient grant/denial response to an access request and has been incorporated in the provision-based access control model (PBAC). Based on PBAC, we propose a context-aware provision-based access control model, capable of dynamic adaptation of access control policy according to the changing context. In particular, the model facilitates the definition of context-aware policies and enriches the access control by enforcing provisional actions in addition to common permissions.

1 Introduction

Pervasive computing sketches a pleasant vision for the future computing environments, as the computing power is provided anywhere, anytime, and using any device. Mobile and stationary devices spread in the environment trying to assist humans in their tasks unnoticeably. The main enabling technology of this vision is context-awareness, i.e. extract, interpret, and use context information and adapt functionality of the system to the current context of use [1]. Due to high heterogeneity and dynamicity in such environments, some requirements are introduced in expressiveness and flexibility of security policies. Therefore, a new trend of research in computer security formed towards designing context-aware security infrastructures and access control models.

It is necessary to define what can be considered as context accurately. Context as defined expressively by Dey [2] is: *"any information that can be used to characterize the situation of an entity."* An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. There are many ways to leverage contextual information in an access control system [3]; two of which are of our interest in this paper. Contextual information can be used by an access control policy to utilize environmental factors specifying how and when the policy is enforced [4]. In addition, the system can express its policy flexibly based on the context of subjects and objects, moving from identity-based towards attribute-based and context-based authorization. Existence of various entities and high dynamicity of future environments urges such flexibility.

On the other hand, ordinary access control models assume a binary decision upon an access request, either grant or deny, which seems to be non-satisfactory in new environments. The idea of provisional authorization states that the user requests will be authorized provided he (and/or the system) takes certain actions prior to authorization of his request [5]. In this manner, a decision upon an access request consists of two components; a permission denoting grant or denial, and a set of provisional actions to be performed before the access decision. Provisional actions, also called provisions, empower policy rules to enforce required actions such as logging and encryption prior to access.

We argue that incorporating the above concepts, namely provisional actions and context-awareness, into a well-structured access control model would be a good candidate to control access in new distributed environments. In this paper, we propose the CA-PBAC as a context-aware provision-based access control model. Access decision in this model is aware of the context, and provisions enable enforcing more powerful and efficient security policies.

The rest of this paper is organized as follows. In section 2, some access control models are surveyed which consider either provisions or context-awareness. Section 3 gives an informal description of CA-PBAC. An access control framework is extended in section 4 to meet our model requirements. In section 5, we present a formal definition of the CA-PBAC model. The formal description prevents any ambiguity in the model and makes implementation and application of the model straightforward. Finally, Section 6 provides a simple system based on CA-PBAC model and explain its function in an access scenario, followed by our conclusion and future works.

2 Related Work

Kudo proposed provision-based access control model (PBAC) to include provisional actions in addition to the common grant or denial decision [6]. He provided a foundation model that addresses fundamental principles of access control, e.g. multiple hierarchies and typical policies for property propagation through hierarchies. The model contains a foundation data set including multiple hierarchical and non-hierarchical sets, and also a relation among these sets that corresponds to access control policy rules. The foundation model decides some properties in response to a property query based on its data set. On the basis of the foundation model, a conventional authorization model and a provisional authorization model were presented to determine binary authorization decision and a set of provisional actions, respectively. PBAC uses a combination of these two authorization models to determine the access decision containing a permission and some provisional actions. Bettini et al. introduced the notion of obligation in addition to provisions [7]. Obligations are those conditions or actions that must be fulfilled by either the users or the system after the access decision is rendered. They proposed a rule-based framework to select the appropriate set of provisions and obligations based on numerical weights assigned to provisions and obligations as well as on semantic relationships among them. Park et al. introduced a family of $UCON_{ABC}$ models for usage control which integrate authorizations, obligations, and conditions [8]. Usage control covers continuity of access control. Similar to provisions, obligations are functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise. The model is abstract in that expresses only its semantics and does not define the authorization procedure.

Han et al. proposed a basic definition of context-sensitive access control [9], which consists of extendible context model, authorization policy model, request model, and the relevant algorithms. Some ordinary contexts like time and location were formally defined. The model is very basic and lacks fundamental features of an access control model such as groups, hierarchies, and conflict resolution. Kouadri et al. introduced contextual graphs as a new modeling approach for specifying context-based policies [10]. It is a variation of decision tree that allows branching based on contextual information, instead of making a decision. The branched paths are recombined after specifying some security actions or more branching. Although it is an expressive way to define context-based policies, management of such policies seems to be complicated.

Many researches are targeted to applying context-awareness to the RBAC model. Al-Kahtani et al. proposed the RB-RBAC model, performing role assignment dynamically based on users' attributes or other constraints on roles [11]. GRBAC [12] incorporates three type of roles; subject roles corresponds to traditional RBAC roles, object roles which are used to categorize objects, and environment roles to capture environmental or contextual information. Zhang et al. proposed DRBAC, a dynamic context-aware access control for pervasive applications [13]. In DRBAC, there is a role state machine for each user and a permission state machine for each role. Changes in context trigger transitions in the state machines. Therefore, user's role and role's permissions are determined according to the context.

3 Narrative Description of CA-PBAC

In order to provide context-awareness, we incorporated a formal specification of contextual information into CA-PBAC. It follows the general form of (<entity>, <context type>, <relator>, <value>) as context predicates. A context predicate describes a *context type* about an *entity* by associating a *value* through a *relator*. For example, (John, location, entering, ConferenceRoom) states that John is entering the conference room, or (Bob, position, is, secretary) expresses Bob's position. The basic idea of such context predicates has been adopted from Gaia project which provides the infrastructure for constructing smart spaces [14]. Therefore, contextual information are expressed by a set of context predicates in the model. The special characteristic of this specification is decomposability, i.e. a predicate can be decomposed into an entity and its context (including a context type, a relator, and a value). Inversely, an entity and a context can be composed to make a contextual information entry.

Hierarchies have been widely employed in access control models. They simplify authorization management by organizing entities and propagating properties through the links among them. The ability to define different propagation strategies [6] or derivations [15] makes hierarchies more efficient. Although CA-PBAC does not provide the generality of the foundation model in [6], it is capable of defining multiple subject group hierarchies and object group hierarchies. There exists a context-assignment function for each hierarchy which assigns a contextual condition including some contexts to each group. A subject requesting an access on an object is mapped to some subject groups according to its context. It is mapped to a subject group, if it's context matches all the contexts in the group contextual condition. Similar context-aware mapping is done for the object.

In practice, it is more likely to construct each hierarchy based on a particular set of context types. The semantics on which hierarchies are defined may vary. This leads to the requirement of different propagation strategies. Selection of an appropriate propagation strategy for each hierarchy, avoids any concern in the model about what actually the mentioned semantics are. In addition to propagation strategies defined in [6], i.e. *"most specific"* and *"path traversing"*, we also consider the *"most general"* strategy. The "most specific" strategy is appropriate to specify exceptions against more general policies. The "path traversing" strategy states that the policies of each ancestor is applicable. We argue that there are some cases in which the "most specific" strategy is not desirable [16], neither is the "path traversing". Actually, the "most general" strategy is useful when there is a need to override more specific policies. As an example, consider specifying a temporary policy to grant authorization to every user in a general group, overriding previously specified denials for more specific groups.

In addition to propagation, hierarchies simplifies conflict resolution based on contextual information. Consider an authorization decision about an access requested by a secretary. Let the first rule state that access is denied if he or she is an employee and the second one state that access is granted if he or she is is a secretary. Clearly, the two rules are in conflict, i.e. a secretary is also an employee. It is common to resolve such conflict by selecting the alternative that states more specific condition, i.e. the second rule. Association of the first rule and the second rule conditions to a parent and a child node in a hierarchy respectively, defining the rules on corresponding nodes, and using "most specific" as propagation strategy avoids such conflicts.

The CA-PBAC model is rule-based. So, the access control policy consists of multiple access control policy rules. Each rule is composed of a group in each subject/object group hierarchy, the requested action on the object, a contextual constraint to limit the context in which the rule is applicable, the permission specified for such action, and the provisions to be executed. Specifying the permission as "NIL" enables the rule to apply some provisions to an access request without defining a permission for such access. That is appropriate to execute common provisions such as logging for accesses regardless of permission result.

Receiving a request from a subject to access an object, a decision including a permission and some provisions is made. Firstly, the subject and the object are mapped to corresponding groups according to their context. Then, according to the policy rules and propagation policies on each hierarchy applicable rules are selected and their permissions are retrieved. The final permission is determined by resolving possible conflict among the retrieved permissions.

In order to determine provisions, a similar procedure is used. However, we consider a difference between propagation of permissions and provisions. Since provisions are obligatory actions, it seems that propagation strategies such as "most specific" are not suitable. For example, "most specific" strategy prevents enforcement of provisions in more general rules even they have no conflicts with the provisions in a selected rule. Actually, we consider full propagation for provisions but restricted to those specified in rules whose permission have no conflict with the decided permission. However, there may be some conflicts among provisions retrieved which are resolved by a domain specific conflict resolution function.

4 Context-Aware Provision-Based Access Control Framework

Illustrated in Figure 1, we suggest a framework to address both data elements and sequences of operations required to provide context-aware provision-based access control. It is based on the PBAC architecture, which itself is a modified version of "Access Control Framework", an international standard [17]. We attempted to use the standard notations according to [17] but adapt their interpretations to meet our model requirements. Extending the standard framework, we introduce two new components. The Contextual Information (CI), which represents all contextual information available to the system. It is assumed that a context infrastructure or simply a context engine provides such information. It includes information that have been stored, sensed from environment, or interpreted from other information. The Access Control Meta Policies (ACMP), which are high-level policies that often used to mediate other policies and are changed less frequently than conventional policies. Conflict resolution and default policies are examples of ACMP.

In this framework, the basic entities and functions involved in an access control scenario are the initiator, the Access Control Enforcement function (AEF), the Access Control Decision Function (ADF), and the target. Initiator, also referred to as *subject* in the access control literature, submits an access request to the system. An access request specifies an operation to be performed on a target, which is referred to as *object* in the access control literature. AEF submits the access request to ADF. ADF decides about the access request using Access Control Decision Information (ADI), Access Control Policy Rules (ACPR), Contextual Information (CI), and Access Control Meta Policies (ACMP). ADI consists of information which is local to access control system, such as group hierarchies. ACPR consists of rules specifying the system current policy. CI is

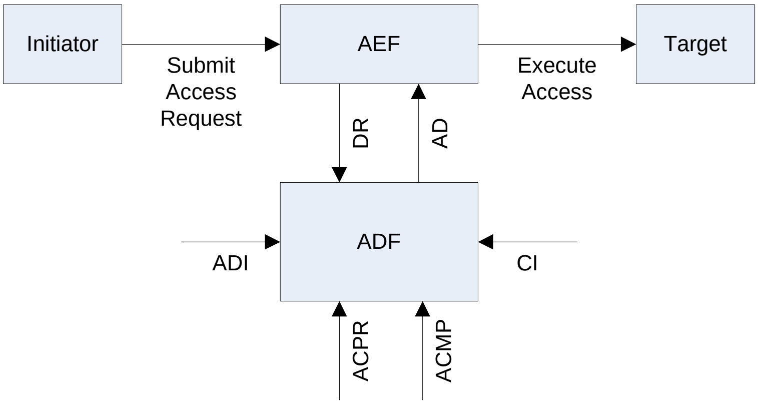

Fig. 1. Context-Aware Provision-Based Access Control Framework

used to interpret both ADI and ACPR. ACMP is used to mediate ACPR and resolve conflicts, in case there is. Based on such inputs, ADF makes the decision consisting of a usual binary decision as well as a set of provisions. Finally AEF enforces the access decision on the target.

5 Context-Aware Provision-Based Access Control (CA-PBAC) Model

In order to have a formal definition of CA-PBAC, we define some basic concepts and then the CA-PBAC data set and access decision function are formally defined. Considering these definitions and also the context-aware provision-based access control framework, a formal definition of our model is presented.

5.1 Basic Definitions

Definition 1 (hierarchy). *Let H be a set forming a partial order, i.e. $\langle H, \leq \rangle$. Formally, H is a hierarchy if and only if $\forall a \in H[\forall x \in H, x \leq a, \forall y \in H, y \leq a[x \leq y \vee y \leq x]]$. Hierarchy H is denoted by $\langle H, \leq^{Tr} \rangle$.*

Definition 2 (Maximal function). *Let H be a hierarchy, i.e. $\langle H, \leq^{Tr} \rangle$ and A be a set such that $A \subseteq H$. The function $Maximal(H, A)$ returns the maximal elements of the hierarchy H restricted to the elements in the set A. Formally, $Maximal(H, A) = \{x \mid x \in A \wedge (\nexists y \in A[y \neq x \wedge x \leq^{Tr} y])\}$.*

Definition 3 (Minimal function). *Let H be a hierarchy, i.e. $\langle H, \leq^{Tr} \rangle$ and A be a set such that $A \subseteq H$. The function $Minimal(H, A)$ returns the minimal elements of the hierarchy H restricted to the elements in the set A. Formally, $Minimal(H, A) = \{x \mid x \in A \wedge (\nexists y \in A[y \neq x \wedge y \leq^{Tr} x])\}$.*

Notation 1 (tuple element) *Let T be a tuple $(t_1, t_2, \ldots, t_k)$. The notation $T.t_i, 1 \leq i \leq k$ corresponds to the element t_i of T.*

Definition 4 (mapping). *A function $f : X_1 \times \ldots \times X_m \rightarrow Y_1 \times \ldots \times Y_n$ is called a mapping if it satisfies:* $\forall\ (x_1, \ldots, x_m) \in X_1 \times \ldots \times X_m[\exists(y_1, \ldots, y_n) \in Y_1 \times \ldots \times Y_m, f(x_1, \ldots, x_m) = (y_1, \ldots, y_n)]$.

5.2 CA-PBAC Data Set: CAPDS

The CA-PBAC model uses a 4-tuple data set $(DS, CI, ACMP, ACPR)$ that formalizes any data structure used in the system.

1. $DS = BDS \cup CDS \cup HDS$; defines required data sets including basic access control data sets, context-related data sets, and hierarchy-related data sets. We introduce each group sets in the following paragraphs.
   - BDS (Basic Data Sets) corresponds to the sets that define the basic components of a provision-based access control model and propagation strategies. It includes
     - $ActSet$ is a set of available actions that can be performed on objects in $ObjSet$.
     - $PvnSet$ is a set of provisions defined according to the application domain.
     - $PrmSet = \{+, -, NIL\}$; is the set of permission results. They correspond to grant, denial, and unspecified permissions, respectively.
     - $PrpStgSet = \{most\_specific, most\_general, path\_traversing\}$; is the set of alternative authorization propagation strategies on a hierarchy. According to a given member h of a hierarchy, "$most\_specific$" means only authorization specified for the most specific member is applicable, "$most\_general$" means only authorization specified for the most general member is applicable, and "$path\_traversing$" means that every authorization specified for members from the root "any" to h are applicable.
   - CDS (Context-related Data Sets) formalize the contextual information used in the system. It includes
     - $EntSet$ is a set of entities that information about their situation can be considered as context in the system. According to Dey's definition of context [2], subjects and objects of the access control system are also considerable as entities that have context. Therefore, $SbjSet$ is a subset of entities ($SbjSet \subseteq EntSet$) that can act actively in the system. Similarly, $ObjSet$ is a subset of entities ($ObjSet \subseteq EntSet$) that can act passively in the system.
     - $CtxTypeSet$ is a set of possible context types, e.g. location, time.
     - $CtxValueSet$ is a set of possible values, e.g. room1, 8pm.
     - $CtxRelatorSet$ is a set of possible relators which relate context types to values, e.g. entering, $\geq$.

    - $CtxSet = CtxTypeSet \times CtxRelatorSet \times CtxValueSet$; forms all possible contexts without binding to a specific entity, e.g. (location, entering, room1).
    - $EntCtxSet = EntSet \times CtxTypeSet \times CtxRelatorSet \times CtxValueSet$; forms all possible contextual information about entities in the system, e.g. (Bob, location, entering, room1) considering Bob in $EntSet$.
  - $HDS = \{SH_1, \ldots, SH_n, OH_1, \ldots, OH_m, SC_1, \ldots, SC_n, OC_1, \ldots, OC_m\}$; (Hierarchy-related Data Sets) composed of subject and object group hierarchies and their corresponding context-assignments.
    - $\forall i, 1 \leq i \leq n, SH_i$ is a set of subject groups forming a hierarchy, i.e. $\langle SH_i, \leq^{Tr} \rangle$. There is a special subject group "*any*" $\in SH_i$ which is the root of the hierarchy.
    - $\forall j, 1 \leq j \leq m, OH_j$ is a set of object groups forming a hierarchy, i.e. $\langle OH_j, \leq^{Tr} \rangle$. There is a special object group "*any*" $\in OH_j$ which is the root of the hierarchy.
    - $\forall i, 1 \leq i \leq n, SC_i : SH_i \rightarrow \mathcal{P}(CtxSet)$; is a mapping which assigns some contexts to each subject group. Note that $SC_i(any)$ is fixed and equals to $\emptyset$. The function result is used to map subjects in $SbjSet$ to subject groups in SH_i.
    - $\forall j, 1 \leq j \leq m, OC_j : OH_j \rightarrow \mathcal{P}(CtxSet)$; is a mapping which assigns some contexts to each object group. Note that $OC_j(any)$ is fixed and equals to $\emptyset$. The function result is used to map objects in $ObjSet$ to object groups in OH_j.

2. $CI \in \mathcal{P}(EntCtxSet)$; corresponds to the set of all contextual information in the current state of the system.
3. $ACMP = (P_{SH_1}, \ldots, P_{SH_n}, P_{OH_1}, \ldots, P_{OH_m}, HsPr, PrmConfRes, DefPrm, PvnConfRes)$; is a tuple corresponding to access control metapolicy. Its elements are defined as:
   - $\forall i, 1 \leq i \leq n, P_{SH_i} \in PrpStgSet$; specifies authorization propagation strategy for the ith subject hierarchy.
   - $\forall j, 1 \leq j \leq m, P_{OH_j} \in PrpStgSet$; specifies authorization propagation strategy for the jth object hierarchy.
   - $HsPr : \{1, \ldots, n+m\} \rightarrow \{SH_1, \ldots, SH_n, OH_1, \ldots, OH_m\}$; is a bijective (one-to-one and onto) function that associates a distinguished subject or object hierarchy to an input rank, establishing a total order among hierarchies. This order affects the selection of applicable policy rules due to different propagation strategies on hierarchies.
   - $PrmConfRes \in \{denials\_take\_precednece, grants\_take\_precedence\}$; specifies which permission overrides when a conflict occurs. "*denials_take_precednece*" means that "−" permission takes precedence over "+" permission, and "*grants_take_precedence*" means that "+" permission takes precedence over "−" permission.
   - $DefPrm \in PrmSet - \{NIL\}$; specifies default permission for an access when no authorization is specified. "+" indicates an open system and "−" corresponds to a close one.

- $PvnConfRes : \mathcal{P}(PvnSet) \rightarrow \mathcal{P}(PvnSet)$; is a domain specific conflict resolution function for provisions.

4. $ACPR \in \mathcal{P}(SH_1 \times \ldots \times SH_n \times OH_1 \times \ldots \times OH_m \times ActSet \times \mathcal{P}(EntCtxSet) \times PrmSet \times \mathcal{P}(PvnSet))$; corresponds to the set of access control policy rules. Each policy rule $r \in ACPR$, structured as $(g_{SH_1}, \ldots, g_{SH_n}, g_{OH_1}, \ldots, g_{OH_m}, act, c, prm, pvns)$, is composed of a member of each subject and object hierarchy, an action, a constraint defined by some contextual information entries, a permission specified for such request, and some provisions.
   Note that when "NIL" used as the permission in a rule it indicates that no permission is specified by the rule and only provisions are considered.

5.3 Access Control Framework Components

Components of the access control framework defined in section 4 are formally defined by the set $\{DR, AD, ADI, CI, ACPR, ACMP\}$ as follows.

- $DR = (sbj, obj, act) \in SbjSet \times ObjSet \times ActSet$; Decision request consists of a subject sbj, an object obj, and an action act requested by the subject to be performed on the object.
- $AD = (prm, pvns) \in (PrmSet - \{NIL\}) \times \mathcal{P}(PvnSet)$; Access decision consists of a permission prm and a set of provisions $pvns$ to be enforced.
- $ADI = (SC_1, \ldots, SC_n, OC_1, \ldots, OC_m)$; Access decision information consists of subject and object context-assignments.
- CI, $ACPR$, and $ACMP$ are exactly as defined in $CAPDS$.

5.4 Access Decision Function

The access decision function is the mapping $ADF : CAPDS \times DR \rightarrow PrmSet \times \mathcal{P}(PvnSet)$. It consists of the following steps:

1. *Context-Aware Mapping Step:*
   In this step, groups of subjects and objects to which $DR.sbj$ and $DR.obj$ can be mapped respectively are selected. To achieve this, each subject group g in the hierarchy SH_i is considered. The subject $DR.sbj$ is mapped to g if it satisfies the contexts defined by the hierarchy context-assignment function SC_i, i.e. $\{(DR.sbj, t, r, v) | (t, r, v) \in SC_i(g)\} \subseteq CI$. Since it is possible to have more than one group from each hierarchy selected for a subject, the set of such subject groups in the hierarchy SH_i are collected in SG_i. Formally, $\forall i, 1 \leq i \leq n, SG_i = \{g \in SH_i \mid \forall (t, r, v) \in SC_i(g)[(DR.sbj, t, r, v) \in CI]\}$. Note that according to the definition of SC_i function, group "$any'' \in SH_i$ will be selected by default for any subject; i.e. $\{(DR.sbj, t, r, v) | (t, r, v) \in SC_i(any)\} = \emptyset \subseteq CI$.
   Analogously, OG_j is formed for each object hierarchy OH_j. Formally, $\forall j, 1 \leq j \leq m, OG_j = \{g \in OH_j \mid \forall (t, r, v) \in OC_j(g)[(DR.obj, t, r, v) \in CI]\}$. Similarly, group "$any'' \in OH_j$ will be selected by default for any object.

2. *Hierarchy Pruning Step:*
In this step, each hierarchy SH_i is pruned to form a sub-hierarchy Q_{SH_i} limited to the elements in SG_i. Formally, $\forall i, 1 \leq i \leq n, Q_{SH_i} = \{s \mid s \in SH_i \wedge \exists g \in SG_i[s \leq^{Tr} g]\}$. Analogously, a sub-hierarchy Q_{OH_j} is formed for each hierarchy OH_j. Formally, $\forall j, 1 \leq j \leq m, Q_{OH_j} = \{o \mid o \in OH_j \wedge \exists g \in OG_j[o \leq^{Tr} g]\}$.
3. *Context-Aware Permission Retrieval Step:*
Firstly, acceptable rules are gathered in a temporary set R_1. A rule is acceptable if its subject/object groups are in the pruned hierarchies resultant from the previous step, its action is equal to the requested action, its permission is definite, and finally its condition is satisfied. Recalling the definition of a single rule $r \in ACPR$, constraint $r.c$ consists of some contextual propositions. Constraint $r.c$ is satisfied if current contextual information include contents of the constraint, i.e. $r.c \subseteq CI$. Next, according to the order defined by $ACMP.HsPr$, the set R_1 is refined considering the propagation strategy of each hierarchy. The resulting set R_{PRM} expresses the applicable rules according to the propagation strategies. Finally, the permission result set PRM is retrieved from the refined rule set. The following algorithm describes this step formally:
   (a) $R_1 = \{r \in ACPR \mid \forall i, 1 \leq i \leq n[r.g_{SH_i} \in Q_{SH_i}] \wedge \forall j, 1 \leq j \leq m[r.g_{OH_j} \in Q_{OH_j}] \wedge r.act = DR.act \wedge r.prm \neq \text{“}NIL\text{”} \wedge r.c \subseteq CI\}$
   (b) $\forall k, 1 \leq k \leq n + m$
      i. $H = HsPr(k)$
      ii. $A = \{r.g_H \mid r \in R_1\}$
      iii. $B = \begin{cases} Maximal(Q_H, A) & \text{if } ACMP.P_H = \text{“}most\_specific\text{”} \\ Minimal(Q_H, A) & \text{if } ACMP.P_H = \text{“}most\_general\text{”} \\ A & \text{if } ACMP.P_H = \text{“}path\_traversing\text{”} \end{cases}$
      iv. $R_1 = \{r \in R_1 \mid r.g_H \in B\}$
   (c) $R_{PRM} = R_1$
   (d) $PRM = \{r.prm \mid r \in R_{PRM}\}$

   Note that $Maximal$ and $Minimal$ functions are defined in section 5.1.
4. *Permission Conflict Resolution Step:*
If no permission could be retrieved, the meta-policy $ACMP.DefPrm$ becomes the permission decision. It is also possible that conflict occurs in the retrieved permission set PRM, i.e. both “+” and “−” permissions are retrieved. In this case, conflict is resolved according to the meta-policy $ACMP.PrmConfRes$. Formally, the access permission is decided as

$$AD.prm = \begin{cases} ACMP.DefPrm & \text{if } |PRM| = 0 \\ p \in PRM & \text{if } |PRM| = 1 \\ \text{“}+\text{”} & \text{if } |PRM| = 2 \wedge \\ & \quad ACMP.PrmConfRes = \\ & \quad \text{“}grants\_take\_precedence\text{”} \\ \text{“}-\text{”} & \text{if } |PRM| = 2 \wedge \\ & \quad ACMP.PrmConfRes = \\ & \quad \text{“}denials\_take\_precedence\text{”} \end{cases}$$

5. *Context-Aware Provision Retrieval Step:*
   The set of applicable rules for retrieving provisions are formally stated by $R_{PVN} = \{r \in ACPR \mid \forall i, 1 \leq i \leq n, r.g_{SH_i} \in Q_{SH_i} \wedge \forall j, 1 \leq j \leq m, r.g_{OH_j} \in Q_{OH_j} \wedge r.act = DR.act \wedge (r.prm = AD.prm \vee r.prm = \text{“}NIL\text{”}) \wedge r.c \subseteq CI\}$. Note that $AD.prm$ was a result of previous step. Therefore the set of provision result is computed as $PVN = \{p \in r.pvns \mid r \in R_{PVN}\}$.
6. *Provision Conflict Resolution Step:*
   The calculated set in the previous step, PVN, may have some conflicting provisions. Such conflicts are domain specific and relevant to semantics of defined provisions. Therefore the domain specific function $ACMP.PvnConfRes : \mathcal{P}(PvnSet) \rightarrow \mathcal{P}(PvnSet)$ resolves these conflicts. Finally, $AD.pvns$ is computed by $ACMP.PvnConfRes(PVN)$.

5.5 CA-PBAC Access Control Model

The CA-PBAC model uses an authorization model formally defined by a 4-tuple $(CAPDS, ADF, DR, AD)$. Based on the framework explained in section 4, CA-PBAC controls accesses by sending an access request as DR into ADF, which itself makes the access decision AD based on the model data set $CAPDS$, and finally enforcing AD. Enforcing AD includes fulfilment of some provisions prior to statement of permission decision.

6 Application Example

In order to illustrate applicability of CA-PBAC, controlling accesses to the internet applications in a university department is given as an example. The required data sets according to section 5.2 are explained very briefly. Basic data sets in BDS are defined as:

$$ActSet = \{use\},$$
$$PvnSet = \{Log, LimitBW, SetMaxSecurity, NotifyTeacher, NotifyManager\}$$

Context-related data sets in CDS are defined as:

$$\begin{aligned}
EntSet &= \{Alice, Bob, MsnMessenger, InternetExplorer, Emule,\\
&\quad RealPlayer, env, network\}\\
SbjSet &= \{Alice, Bob\}\\
ObjSet &= \{MsnMessenger, InternetExplorer, Emule, RealPlayer\}\\
CtxTypeSet &= \{occupation, position, resources, type, traffic, time\}\\
CtxValueSet &= \{student, employee, professor, staff, class, laboratory, internet,\\
&\quad P2P, messenger, multimedia, browser, low, launch\_time\}\\
CtxRelatorSet &= \{is, in, not\_in\}
\end{aligned}$$

Figure 2 illustrates subject and object hierarchies in the system, namely SH_1, SH_2, and OH_1. For instance according to Figure 2.a, we have $EMP \leq^{Tr} PROF$

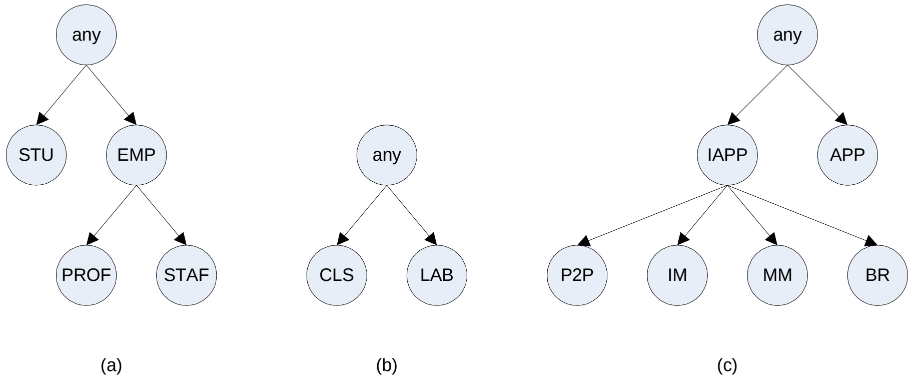

Fig. 2. Example hierarchies in a university department: a)SH_1 b)SH_2 c)OH_1

in SH_1. The formal definition of each hierarchy and its context-assignment function follows:

$SH_1 = \{any, STU, EMP, PROF, STAF \mid any \leq^{Tr} STU \wedge any \leq^{Tr} EMP \wedge$
$\quad EMP \leq^{Tr} PROF \wedge \; EMP \leq^{Tr} STAF \; \}$
$SC_1 = \{(any, \emptyset), \; (STU, \{(occupation, is, student)\}),$
$\quad (EMP, \{(occupation, is, employee)\}), (PROF, \{(position, is, professor)\}),$
$\quad (STAF, \{(position, is, staff)\}) \; \}$
$SH_2 = \{any, CLS, LAB \mid \ldots\}$
$SC_2 = \{(any, \emptyset), \; (CLS, \{(location, in, class)\}), \; (LAB, \{(location, in, laboratory)\}) \; \}$
$OH_1 = \{any, IAPP, APP, P2P, IM, MM, BR \mid \ldots\}$
$OC_1 = \{(any, \emptyset),$
$\quad (IAPP, \{(resources, include, internet)\}),$
$\quad (APP, \{(resources, not\_include, internet)\})$
$\quad (P2P, \{(resources, include, internet), (type, is, P2P)\}),$
$\quad (IM, \{(resources, include, internet), (type, is, messenger)\}),$
$\quad (MM, \{(resources, include, internet), (type, is, multimedia)\}),$
$\quad (BR, \{(resources, include, internet), (type, is, browser)\})\}$

The CI component should be considered when an access request is made. Elements of the $ACMP$ tuple containing meta-policies is defined as:

$P_{SH_1} =$ "$most\_specific$", $\; P_{SH_2} =$ "$most\_specific$", $\; P_{OH_1} =$ "$path\_traversing$",
$HsPr = \{(1, SH_1), (2, SH_1), (3, OH_1)\}, \; PrmConfRes =$ "$denials\_take\_precedence$",
$DefPrm =$ "$+$", $\; PvnConfRes = \{(pvns, pvns) \mid pvns \in \mathcal{P}(PvnSet)\}$

Finally, the system has following policy rules defined in $ACPR$:

$ACPR = \{r_1, r_2, r_3, r_4, r_5, r_6\}$

$$\begin{aligned}
&r_1 = (STU, any, MM, use, \{(network, traffic, is, low)\}, +, \{LimitBW(128kbps)\}),\\
&r_2 = (STU, CLS, IAPP, use, \{\}, -, \{NotifyTeacher\}),\\
&r_3 = (STU, CLS, any, use, \{\}, +, \{log\}),\\
&r_4 = (EMP, any, IM, use, \{\}, NIL, \{log\}),\\
&r_5 = (EMP, any, IAPP, use, \{\}, +, \{SetMaxSecurity\}),\\
&r_6 = (STAF, any, IM, use, \{(env, time, not\_in, launch\_time)\}, -, \{NotifyManager\}).
\end{aligned}$$

r_1 states that if the network traffic is low students are allowed to use multimedia applications, provided that their bandwidth is limited to 128kbps. r_2 states that students in class are not allowed to use internet applications, but their teacher is notified of their request. r_3 states that students can use anything (here any applications), but prior to use it is logged. r_4 states that employees' request of using instant messaging applications will be logged. r_5 states that employees are allowed to use internet applications, provided that the security level of their application is set to maximum. r_6 states that staff are not allowed to use instant messaging applications out of launch time, however their manager is notified of such request.

Now consider the following scenario. Alice, who is a student, wants to use Real Player application to watch news online in class. The decision request $DR = (Alice, RealPlayer, use)$ is submitted to the ADF function. The contextual information CI at the time of the request consists of

$$\begin{aligned}
CI = \{&(Alice, occupation, is, student),\ (Alice, location, in, class),\\
&(RealPlayer, resources, include, internet),\ (RealPlayer, type, is, multimedia),\\
&(network, traffic, is, low),\ \ldots\}
\end{aligned}$$

The following steps are taken in the ADF:

1. The result of mappings becomes $SG_1 = \{any, STU\}$, $SG_2 = \{any, CLS\}$, and $OG_1 = \{any, IAPP, IM\}$
2. Pruning each hierarchy according to the above sets results to $Q_{SH_1} = \{any, STU \mid any \leq^{Tr} STU\}$, $Q_{SH_2} = \{any, CLS \mid any \leq^{Tr} CLS\}$, and $Q_{OH_1} = \{any, IAPP, IM \mid any \leq^{Tr} IAPP \wedge IAPP \leq^{Tr} IM\}$
3. Policy rule r_1 is acceptable because $STU \in Q_{SH_1} \wedge any \in Q_{SH_2} \wedge MM \in Q_{OH_1} \wedge DR.act =$ "use" $\wedge$ "$+$" $\neq$ "NIL" $\wedge \{(network, traffic, is, low)\} \subseteq CI$. Similarly, r_2 and r_3 are selected. Therefore, in step (a) of the algorithm $R_1 = \{r_1, r_2, r_3\}$. The following table shows the execution of loop at step (b):

k	R_1	H	A	$ACMP.P_H$	B
1	$\{r_1, r_2, r_3\}$	SH_1	$\{STU\}$	$most\_specific$	$\{STU\}$
2	$\{r_1, r_2, r_3\}$	SH_2	$\{any, CLS\}$	$most\_specific$	$\{CLS\}$
3	$\{r_2, r_3\}$	OH_1	$\{any, IAPP, MM\}$	$path\_traversing$	$\{any, IAPP, MM\}$
4	$\{r_2, r_3\}$	-	-	-	-

The applicable rules R_{PRM} becomes $\{r_2, r_3\}$ and the retrieved permission set PRM becomes $\{-, +\}$.

4. Since there is a permission conflict and $ACMP.PrmConfRes$ equals to "$denials\_take\_precedence$", $AD.prm$ becomes "$-$".
5. The set of applicable rules to retrieve provisions R_{PVN} becomes $\{r_2\}$. So the set of provisions becomes $PVN = \{NotifyTeacher\}$.
6. Since $ACMP.PvnConfRes$ is defined as a reflexive function, the final provisions becomes:
$AD.pvns = ACMP.PvnConfRes(\{NotifyTeacher\}) = \{NotifyTeacher\}$.

Eventually, the access decision is composed from results of steps 4 and 6, i.e. $AD = (\text{``} - \text{''}, NotifyTeacher)$. Alice is denied to use Real Player application and also her teacher is notified of her request.

Consider another scenario; A staff named Bob requests to use MSN messenger application in launch time. CI at the time of the request consists of

$$
\begin{aligned}
CI = \{&(Bob, occupation, is, employee),\ (Bob, position, is, staff),\\
&(MsnMessenger, resources, include, internet), (MsnMessenger, type, is, messenger),\\
&(env, time, in, launch\_time),\ \ldots\}
\end{aligned}
$$

Following the steps in ADF, it is decided that Bob is allowed to use MSN messenger application provided its security level is set to maximum and the access is logged.

7 Conclusion

In this paper, we extended a standard access control framework to include contextual information, access control meta policies, and provisions. Based on the framework, we proposed and formally specified our CA-PBAC model which enables definition of context-aware policy and enriches the access control by enforcing provisions in addition to common permissions. Context is incorporated into group hierarchies using contextual condition assignment to each group. Actually, hierarchies are formed over contextual groups to which subjects and objects are mapped. Constructing multiple hierarchies, each probably based on a particular set of context types, allows high flexibility in defining subjects and objects based on their context. In addition, different propagation strategies enable different hierarchical definition semantics. Moreover, specification of contextual constraints in policy rules controls the applicability of rules according to the context, i.e. the overall policy is dynamically adapted to the current context.

Enhancements to the model can be considered as future works. These include 1) separation of the access control model from the context model, 2) contextual provisions which are determined according to the context, and 3) conflict resolution based on the context of rules.

References

1. Korkea-aho, M.: Context-aware applications survey. Technical report, Helsinki University of Technology (2000)

2. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing **5**(1) (2001) 4–7
3. Thomas, R.K., Sandhu, R.S.: Models, protocols, and architectures for secure pervasive computing: Challenges and research directions. In: 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), Orlando, FL, USA (2004) 164–170
4. McDaniel, P.D.: On context in authorization policy. In: 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), Villa Gallia, Como, Italy, ACM (2003)
5. Jajodia, S., Kudo, M., Subrahmanian, V.S.: Provisional authorizations. In: 1st Workshop on Security and Privacy in E-Commerce, Athens, Greece (2000)
6. Kudo, M.: Pbac: Provision-based access control model. International Journal of Information Security **1**(2) (2002) 116–130
7. Bettini, C., Jajodia, S., Sean Wang, X., Wijesekera, D.: Provisions and obligations in policy management and security applications. In: 28th International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China, Morgan Kaufmann (2002) 502–513
8. Park, J., Sandhu, R.S.: The ucon$_{\mathrm{abc}}$ usage control model. ACM Transactions on Information and System Security **7**(1) (2004) 128–174
9. Han, W., Zhang, J., Yao, X.: Context-sensitive access control model and implementation. In: Fifth International Conference on Computer and Information Technology (CIT 2005), Shanghai, China, IEEE Computer Society (2005) 757–763
10. Kouadri Mostéfaoui, G., Brézillon, P.: Modeling context-based security policies with contextual graphs. In: 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), Orlando, FL, USA, IEEE Computer Society (2004) 28–32
11. Al-Kahtani, M.A., Sandhu, R.S.: A model for attribute-based user-role assignment. In: 18th Annual Computer Security Applications Conference (ACSAC 2002), Las Vegas, NV, USA, IEEE Computer Society (2002) 353–364
12. Moyer, M.J., Ahamad, M.: Generalized role-based access control. In: 21st International Conference on Distributed Computing Systems. (2001) 391–398
13. Zhang, G., Parashar, M.: Context-aware dynamic access control for pervasive applications. In: Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, USA (2004)
14. Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: A middleware infrastructure for active spaces. IEEE Pervasive Computing **1**(4) (2002) 74–83
15. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorizations. In: IEEE Symposium on Security and Privacy, Oakland, CA, USA, IEEE Computer Society (1997) 31–42
16. Dunlop, N., Indulska, J., Raymond, K.: Methods for conflict resolution in policy-based management systems. In: 7th IEEE International Enterprise Distributed Object Computing Conference, Brisbane, Australia, IEEE Computer Society (2003) 98–109
17. ITU-T: Security Frameworks for Open Systems: Access Control Framework. ITU-T Recommendation X.812. (1995)