Conflict Detection and Resolution in Context-Aware Authorization

Amirreza Masoumzadeh, Morteza Amini, and Rasool Jalili
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran *
{masoumzadeh@ce., m_amini@ce., jalili@}sharif.edu

Abstract

Pervasive computing environments introduce new requirements in expressiveness and flexibility of access control policies which are almost addressable leveraging contextual information. Although context-awareness augments the expressiveness of policies, it increases the probability of arising conflicts. Generally, context-aware authorizations are defined using some contextual constraints on the involved entities in an access request. Accordingly, principles like "more specific overrides", which are employed to resolve possible conflicts, are required to consider the contextual constraints. In this paper, we formalize the use of context constraints in a typical context-aware multi-authority policy model; each authority is capable of defining an expressive conflict resolution policy leveraging context-based precedence establishment principles. Based on the policy model, we propose a comprehensive graph-based approach to resolve conflicts. The strength of the approach is that conflict detection which requires context-based inference is almost done statically and resolution is left for run-time.

1. Introduction

Evolution of distributed systems, mobile computing environments, and moving toward pervasive computing environments introduced new security and access control requirements [14], including expressiveness and flexibility of security policies. Leveraging contextual information can address some of these requirements. Context as defined expressively by Dey [3] 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. In an access control system, authorization policies specify which activities should be permitted or forbidden. Classic access control models express their policies using the notions of subjects, objects, and rights [11]. Several works addressed context-aware access control models or security infrastructures such as [5, 2]. We also proposed a context-aware provisional access control model that decides some required provisional actions in addition to common binary access decision according to the context [8]. The central idea of all these models is expressing subjects, objects, groups, roles, etc. through context expressions.

Authorization conflicts arise where two or more different policies both permit and forbid an access in a situation. Although context-awareness highly augments the expressiveness of authorization policies, it increases the probability of conflicts among different policies. A practical solution for resolving conflicts is establishing a precedence among conflicting policies. This can be performed by manual assignment of specific priority to each conflicting policy. However, precedence establishment is more preferred to be done automatically because manual assignment may be cumbersome and impractical in real-world situations. Several principles have been suggested for establishing precedence automatically such as specific overrides general policy (more specific overrides), newer overrides older policy, negative/positive policy takes precedence, and higher authority overrides lower authority [7, 4]. Practically, each principle has its specific application and is useful in a particular situation.

To the best of our knowledge, precedence establishment principles, such as "more specific overrides", has been used to resolve conflicts in context-aware authorization systems only considering a few aspects of context. Our main contribution in this paper is the definition and employment of precedence establishment principals in a context-aware manner, particularly context-based specificity relation among context-aware authorizations. To achieve that, context-aware authorizations are formalized in a rule-based policy model which tends to serve as a common basis for other context-aware authorization models. Since central-

*This work is partially supported by Iran Telecommunication Research Center (ITRC) under grant No. 500/8478.

ized authorization is not feasible for large distributed systems [12] and many context-aware environments are inherently distributed, the policy model supports multi-authority to enable decentralized authorization.

We propose a comprehensive conflict detection and resolution method which supports flexible conflict resolution policies capable of employing different context-based precedence establishment principals. The formalization of the method is expressed through graphs to make it more comprehensible. Our approach considers all the conflicting authorizations together. This is due to the possibility of relationship among conflicts [7], and yielding different results when the sequence of pairwise conflict resolution changes.

The remainder of this paper is organized as follows. Some major works regarding conflict detection and resolution are surveyed in section 2. In section 3, a typical context-aware authorization policy model leveraging a formal definition of context is presented. The context-aware conflict detection and resolution scheme based on the mentioned policy model is proposed in section 4. Section 5 concludes the paper.

2. Related Work

Lupu et al studied conflicts in authorization and obligation policies [7] and defined specificity related to domain nesting as an equivalent for "more specific overrides" principle. Authorization models such as [6] handle principles like "more specific overrides" through different derivation schemes along subject hierarchies.

Ruan et al addressed conflict resolution in presence of authorization delegation through a formal graph based framework [10].

Dunlop et al considered four possible approaches for the process of conflict resolution [4]. In pessimistic approach both potential and actual conflicts are resolved at compile-time, while sin opposite, optimistic approach does the all at run-time. In the balanced alternative, actual conflicts are resolved at compile-time and potential ones remain to run-time. Another alternative is deciding to resolve each conflict individually based on its likelihood of occurring and cost of resolution.

Syukur et al investigated policy conflict resolution in pervasive computing environments [13]. They discussed different timing strategies for conflict detection: static, reactive, proactive, and predictive. However their conflict resolution techniques seems too limited.

Al-Kahtani et al used the notion of dominance between authorization rules in their attribute-based user-role assignment model [1]. Dominance in their work is somehow the reverse notion of context-based specificity in our approach. However, it supports a limited concepts such as ordinal attributes and is used to induce seniority among authorizations in order to construct the induced role hierarchies.

3. Context-Aware Authorization Policy Model

In this section, we provide a typical authority policy model in order that the conflict resolution scheme be general enough to support a wide range of context-aware authorization systems. The model is rule-based in which authorization rules are defined in terms of some constraints on the contextual information.

3.1. Context Definitions

In order to formalize the use of contextual information in the model, some definitions regarding context are provided.

Definition 1 (Context Predicate) *A* context predicate *is a 4-ary tuple* $\langle subject, type, relater, object \rangle$.

Subject is an entity with which the context is concerned, *type* refers to the type of context the predicate is describing, and *object* is the value related to the subject through the *relater*. For example, $\langle Bob, location, entering, room_A \rangle$ states that Bob is entering the room_A, and $\langle John, position, is, secretary \rangle$ expresses John's position. The basic idea of such context predicates has been adopted from Gaia project which provides the infrastructure for constructing smart spaces [9]. It is supposed that there is a means to verify a context predicate in the actual environment. This can be performed by a context infrastructure or a separate component in the system architecture usually called context inference engine. A context predicate is satisfied if it is verified as correct.

Definition 2 (Context Constraint) *A* context constraint *is a set of context predicates.*

In fact, a context constraint is interpreted as logical conjunction of its predicates. It is satisfied if all its context predicates are satisfied. An empty context constraint, which is denoted by $\emptyset$, is satisfied by default. A context constraint can descriptively express a situation based on context information. For instance, the following constraint expresses that Bob is entering the conference room in which a presentation is carried out:

$$\{\ \langle Bob, location, entering, conf\_room \rangle, \\ \langle conf\_room, social\_activity, is, presentation \rangle \}$$

Since one of the most preferable principles to establish precedence is "more specific overrides", we provide the definition of specificity based on context predicates and constraints. For example, a predicate stating that *the age of the* *user must be over 30* should be considered more specific than a predicate expressing *the age of the user must be over 20*.

Definition 3 (More Specific Context Predicate) *Context predicate p_1 is* more specific *than context predicate p_2, denoted by $p_1 <_{MS} p_2$, if they both have equal subject and type, and p_2 is inferable by p_1; i.e. wherever p_1 is satisfied, p_2 is also satisfied.*

Several approaches can be employed to perform the required inference, i.e. $p_1 \rightarrow p_2$. A simple approach is using a knowledge base and a rule-based inference mechanism. Using the following rule could easily yield the previous example, i.e. $\langle user, age, >, 30\rangle <_{MS} \langle user, age, >, 20\rangle$:

$$\langle S, T, >, O_1\rangle \ \wedge \ O_1 > O_2 \rightarrow \langle S, T, >, O_2\rangle$$

Definition 4 (More Specific Context Constraint) *context constraint c_1 is* more specific *than context constraint c_2 regarding a tuple $\langle s, t\rangle$, denoted by $c_1 <^{s,t}_{MS} c_2$, if c_1 contains a predicate with subject s and type t, and either c_2 does not have a predicate with the same subject and type or the c_1's predicate is more specific than c_2's.*

For instance, consider the following two constraints and the specificity relations between them:

$$C_1 = \{(user, age, >, 20), (user, location, in, class\_A)\}$$
$$C_2 = \{(user, age, \geq, 30)\}$$
$$C_2 <^{user,age}_{MS} C_1 \quad , \quad C_1 <^{user,location}_{MS} C_2$$

3.2. Policy Model

Authorization policy is actually a collection of authorization rules briefly termed authorizations.

Definition 5 (Authorization) *An* authorization *is a tuple $\langle sign, condition\rangle$.*

The *sign* can be either "$+$" to state a positive authorization or "$-$" to declare a negative one. The *condition* is a context constraint that specifies the situation at which the authorization is active. Notations *SBJ*, *OBJ*, and *ACT* may be used to address typical access request components as entities of context predicates. For instance, the following authorization denies any access from a user to a confidential or higher classified document remotely:

$$\langle - , \{ \langle SBJ, connection\_type, is, remote\rangle, \langle OBJ, type, is, document\rangle, \langle OBJ, class, \geq, confidential\rangle \} \rangle$$

The condition part of the authorizations might seem somewhat limited in the way that it does not allow disjunctions. However, since all policy authorizations whose condition is satisfied are active and enforceable in an access situation, the enforced policy is considered as the disjunction of all active authorizations. In order to state an authorization with a disjoint condition $A \vee B$, two authorizations, one with condition A and one with condition B and both with the same sign, should be defined. The mentioned limitation facilitates conflict resolution based on context which in its absence can be quite complicated and costly.

Since many context-aware environments such as pervasive computing environments are inherently distributed, centralized authorization management is not practical. So, in order to support decentralized authorization, the policy model supports multiple authorities. Each authority is capable of defining policy for a restricted virtual space called *authority space*.

Definition 6 (Authority Space) *An* Authority space *is defined and limited by a context constraint.*

Within an authority space, an authority defines its policy using authorization rules. Based on the current context, an authority space restricts the subjects, objects, and actions on which policy can be defined or the situation when it is applicable. Actually, an authorization defined by the authority is enforceable when both authority space constraint and authorization condition are met.

In this model, the authorities are organized as follows. There is a *global authority* which its authority space has no constraint. An authority can create several sub-authorities within a more restricted authority space than itself has; the creator's authority space constraint is enclosed in the constraint of the new authority space. Actually, an authority delegates the policy specification responsibility to the sub-authority restricted to its authority space constraint. Note that the authority spaces of sub-authorities are not necessarily isolated; they can overlap each other. In this manner, authorities form a tree structure such that its root is the aforementioned global authority, and each authority is the parent of the authorities which it has created. This way, the model provides authorities with maximal independency in specification of policy. Also the distribution of conflict detection and resolution would be facilitated. Figure 1 illustrates an authority hierarchy. Note that although authority space of the three users is contained in the authority space of the presenter, they are considered as sub-authorities of the room manager.

4. Conflict Detection and Resolution

In the rule-based authorization policy model of section 3, conflicts may arise among different authorizations specified by an authority.

Definition 7 (Conflicting Authorizations) *Two or more authorizations are* conflicting *in a situation if their condition elements are satisfied and they have conflicting sign elements; i.e. some have "$+$" and others have "$-$" sign.*

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

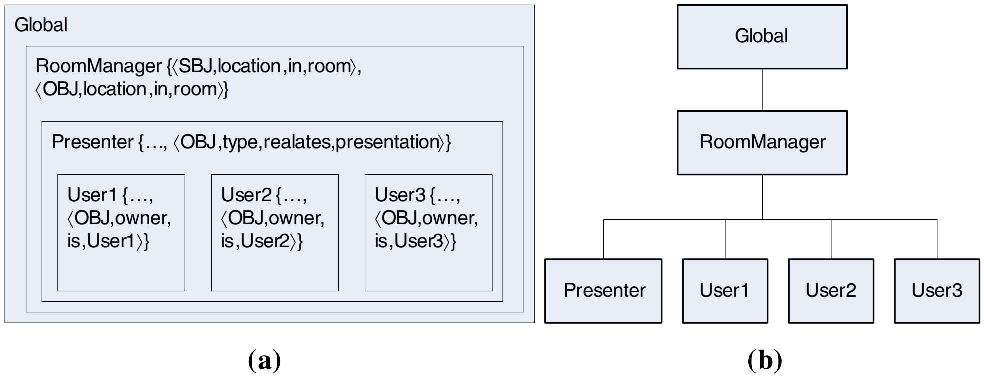

Figure 1. Sample authority spaces (a), and their corresponding authority hierarchy (b)

Furthermore, an access request may be in multiple authority spaces, and different involved authorities may have conflicting decisions. Organizing authorities in a tree structure, as described in section 3.2, provides a reasonable way to resolve the conflicts among the different authorities: the decision of a parent authority overrides the decisions of its children. Thus an authority can resolve the conflicts among its sub-authorities and its own authorization rules to make its final decision. The resolution is done according to resolution policies which are defined by the authority based on the aforementioned principles in a context-aware manner. The resolution is formally defined in section 4.2.

The principles for establishing precedence among authorizations may seem inappropriate for resolving conflicts among sub-authorities. Therefore, in addition to use of those principles, each authority can define a context-aware seniority relation among its sub-authorities.

Definition 8 (Seniority Rule) *A* seniority rule *is a triple* $\langle condition, SA_1, SA_2 \rangle$.

if the context constraint *condition* is satisfied, then sub-authority SA_1 is senior to sub-authority SA_2; i.e. SA_1's decision overrides SA_2's. Note that leaving the condition of a rule empty makes it active constantly and imposes a strict seniority. The seniority relation in an authority is comprised of multiple seniority rules.

4.1. Conflict Detection

We express the formalism of our approach through graphs to make it more comprehensible. Forming a *potential conflict graph*, an authority detects potential conflicts and investigates possible precedence relations among its authorizations. As mentioned in section 3.2 an authority is responsible for resolving conflicts among its sub-authorities in addition to its defined authorizations. In order to increase the consistency of the graph, in the decision situation for a specific access request, a sub-authority is considered as a single authorization: context constraint of sub-authority space as its condition, and its determined decision for the request as its sign.

Definition 9 (Potential Conflict Graph) Potential conflict graph *is a multi graph in which each vertex corresponds to an authorization or a sub-authority. Each edge in the graph represents an overriding relation between two vertices and is labeled with the relations's symbol.*

Possible relations corresponding to an edge from vertex a_1 to a_2 include

- $<_{MS}^{s,t}$, for different values of subject s and type t, if and only if a_1 and a_2 have conflicting signs and $a_1.condition <_{MS}^{s,t} a_2.condition$
- NoP, if and only if a_1 is a negative authorization and a_2 is a positive one
- $>_S$, if and only if there is a seniority rule $\langle c, a_1, a_2 \rangle$ and condition c is satisfied

Overriding relations, presented by labels, can be extended to include more details or complicated context of authorizations such as relation about time of authorization definition. In practice, since there can be various relations of type $<_{MS}^{s,t}$, considering different values for subject s and type t, it is better to limit the subject and types on which the relation is definable.

Also, note that a reverse relation for each relation is supposable. For instance, for each *more specific* relation we can assume a *more general* relation in reverse. Those relations are not mentioned in the graph due to redundancy issues; whenever needed, the reverse of relation α is denoted by α^{-1}. For instance, relation $<_{MS}^{SBJ,location^{-1}}$ states a more general location context for subject, or relation NoP^{-1} expresses that a positive authorization overrides a negative one.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

4.2. Conflict Resolution

In order to define in an expressive how the conflicts in an authority space should be dealt with, the notion of *resolution policy* is introduced.

Definition 10 (Resolution Policy) Resolution policy *is a subset of possible relations' symbols in the potential conflict graph or their reverses.*

Resolution policy can be considered as a group of precedence relations; if all relations corresponding to symbols in a resolution policy exist from authorization a_1 to a_2, then a_1 overrides a_2. For instance, resolution policy $\{<_{MS}^{SBJ,location}, <_{MS}^{OBJ,location}\}$ states that for each two authorizations, if the first authorization specifies more specific location context condition for subject and object then it overrides the second authorization. An advantage of utilization of different relations in a resolution policy is the ability of combining different precedence establishment principles. For example, $\{<_{MS}^{ACT,type}, NoP\}$ expresses that negative authorizations with more specific action types precede positive authorizations.

An authority is capable of expressing its conflict resolution policy using a *resolution policy sequence*.

Definition 11 (Resolution Policy Sequence) Resolution policy sequence *is a total order of some resolution policies. The last resolution policy in the sequence must be either* $\{NoP\}$ *or* $\{NoP^{-1}\}$.

The conflict resolution is a step by step process. Sequentially, at each step, a resolution policy from resolution policy sequence is selected to resolve remaining conflicts; until no conflict remains. As defined, the last resolution policy must state that either negative or positive authorization precedes. That way, it is assured that existing conflicts in an authority space are resolved eventually.

In an actual conflict situation, an *actual conflict graph* is constructed from the potential conflict graph.

Definition 12 (Actual Conflict Graph) Actual conflict graph *is a multi graph extracted from the potential conflict graph which is composed of vertices corresponding to conflicting authorizations at the actual conflict situation and their corresponding edges; i.e. by eliminating those vertices whose corresponding authorizations are not in conflict.*

Actual conflict graph is then pruned according to the resolution policy sequence; Sequentially a resolution policy is selected and the corresponding *elimination graph* is constructed.

Definition 13 (Elimination Graph) *Let* AG *be an actual conflict graph and* R *be a resolution policy. The* elimination graph $EG_{AG,R}$ *is a single graph in which*

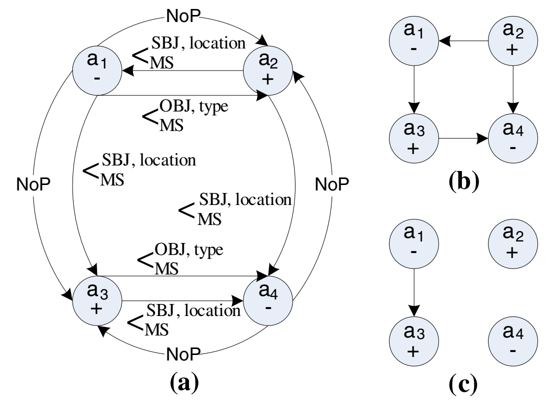

Figure 2. Sample actual conflict graph (a), and its corresponding elimination graphs regarding resolution policy $\{<_{MS}^{SBJ,location}\}$ (b), and resolution policy $\{<_{MS}^{SBJ,location}, NoP\}$ (c)

- *there is a vertex for each vertex in AG, and*
- *there is an unlabeled edge from vertex* a_1 *to vertex* a_2 *if an edge exists in AG*
  - *from* a_1 *to* a_2*, for every relation's symbol in* R *with the same symbol as its label, and*
  - *from* a_2 *to* a_1*, for every reverse relation's symbol in* R *with the same symbol as its label.*

Constructing the elimination graph, the actual graph is pruned by omitting vertices corresponding to non-root vertices in the elimination graph. The enforcement of resolution policies is continued until no conflict exists. Enforcing last resolution policy according to the definition, eventually resolves either negative or positive authorization.

Let us illustrate the approach through a simple example. Consider figure 2.a as an actual conflict graph constructed in a conflict situation by omitting non-conflicting vertices. If we use the resolution policy $\{<_{MS}^{SBJ,location}\}$, the elimination graph in figure 2.b would be constructed. Using this graph to prune the actual conflict graph, all non-root vertices in elimination graph, i.e. a_1, a_3, and a_4 must be deleted from the actual graph. If we use resolution policy $\{<_{MS}^{SBJ,location}, NoP\}$ instead, the elimination graph in figure 2.c would be constructed. Accordingly, only vertex a_3 must be omitted from the actual conflict graph.

4.3. Timing Strategy

Factually speaking, the conflict resolution process is a computationally intensive and time consuming task. Conflicts can be detected and resolved either statically at compile time, or dynamically at run time. But due to its cost, it is more preferable to be done statically [4]. Conflict resolution in a context-aware authorization system is even more complicated; determination of context specificity relations among authorizations requires inference power which is computationally intensive.

The strength of the proposed scheme in this paper is that conflict detection can be performed almost statically, and the resolution process is left for run time. The frequency of policy modification is generally far less than the frequency of arising conflicts in a context-aware policy at run time. Therefore a potential conflict graph is maintained for each authority which is altered by modification of the policy; i.e. adding, deleting, and updating authorizations. Almost all overriding relations among authorizations and sub-authorities which are computationally intensive due to inference requirement are determinable statically. Exceptions are those relations pertaining to the sign, e.g. NoP, between two sub-authorities or between a sub-authority and an authorization. Undoubtedly, those relations should be checked at run time if necessary.

4.4. Algorithms

In this section we provide detailed algorithms to implement the conflict resolution scheme. We also provide the computational complexity of the algorithms in which N is the number of vertices in the conflict graph, and L is the number of possible precedence establishment relations. Actually, N is the number of authorization rules that an individual policy administrator considers and probably is not very high. However, the number of possible edge labels L should be limited for the scheme to be practical.

Algorithm 1 UpdatePotentialGraph_AddAuth(PG, A)

Input: potential conflict graph $PG = (V, E)$, newly added authorization A
Output: updated potential graph PG

$V \leftarrow V \cup \{A\}$
for each $v \in V \backslash \{A\}$ **do**
  for each $\alpha \in PossibleRelationSymbols$ **do**
    if $v.sign \neq A.sign$ **then**
      if $v.condition\ \alpha\ A.condition$ **then**
        $E \leftarrow E \cup \{(v, A, \alpha)\}$
      end if
      if $A.condition\ \alpha\ v.condition$ **then**
        $E \leftarrow E \cup \{(A, v, \alpha)\}$
      end if
    end if
  end for
end for

Potential conflict graph can be maintained statically. Algorithm 1 demonstrates how to update the potential conflict graph PG when a new authorization A is defined by the authority. First, a vertex corresponding to the new authorization is added to PG. Then, for each vertex of PG except A, existence of different types of relations between the vertex and A is checked. The complexity of the algorithm is $O(LN).O(I)$ where $O(I)$ is the complexity of inferring a context-based relation.

Algorithm 2 CreateElimGraph(AG, R)

Input: $AG = (V, E)$, and resolution policy R
Output: $EG_{AG,R} = (V', E')$

$V' \leftarrow V$
$E' \leftarrow \emptyset$
for each two different vertices a and b in V **do**
  $L_{ab} \leftarrow \{l \mid (a, b, l) \in E\}$
  $L_{ba} \leftarrow \{l \mid (b, a, l) \in E\}$
  $Q_r \leftarrow \{\alpha \mid \alpha^{-1} \in R\}$
  $Q \leftarrow R \setminus Q_r$
  if $Q \subseteq L_{ab} \wedge Q_r \subseteq L_{ba}$ **then**
    $E' \leftarrow E' \cup \{(a, b)\}$
  end if
  if $Q \subseteq L_{ba} \wedge Q_r \subseteq L_{ab}$ **then**
    $E' \leftarrow E' \cup \{(b, a)\}$
  end if
end for

Algorithm 2 is used to create a filtered graph from the actual conflict graph AG based on the resolution policy R. It first copies the vertices of AG to $FG_{AG,R}$ and initializes the edges of $FG_{AG,R}$ to null. Then, it checks for every two possible vertices in AG if one of them overrides the other according to the resolution policy R. The complexity of the algorithm is $O(LN^2)$

Algorithm 3 ResolveConflicts(AG, RS)

Input: actual conflict graph $AG = (V, E)$, and resolution policy $RS = (R_1, R_2, \ldots, R_m)$
Output: resolved sign D

for $i \leftarrow 1$ to m **do**
  if $E = \emptyset$ **then**
    break
  end if
  $EG_{AG,R_i} = (V', E') \leftarrow$ CreateElimGraph(AG, R_i)
  $V \leftarrow \{b \in V \mid \nexists (a, b) \in E'\}$
  $E \leftarrow \{(a, b, l) \in E \mid a \in V \wedge b \in V\}$
end for
$D \leftarrow V[0].sign$

Algorithm 3 resolves the conflicts in an actual conflict graph AG using resolution policy sequence RS and results the final decision D. The algorithm sequentially selects the next resolution policy and uses it to construct the elimination graph EG. It then keeps only those vertices of AG

whose corresponding vertices in EG have no input edges; i.e. those authorizations which are not overridden by others according to the current resolution policy. The edges corresponding to deleted vertices are also removed. The iteration stops when no conflict remains among authorizations; i.e. AG has no edges. Note that an edge in AG means the existence of a conflict. Finally, the sign of the authorization corresponding to one of the vertices is returned as the result. The complexity of the algorithm is $O(mLN^2)$ where m is the number of resolution policies in the resolution policy sequence RS.

Note that algorithm 3 resolves only conflicts in an authority. The overall conflict resolution is a recursive process in which an authority requires determination by its involved children authorities and resolves the possible conflicts; The process is continued until no conflict exists among involved authorities. Since the depth of the authorization hierarchy roughly corresponds to the administration levels which is restricted in nature, it can be inferred that the overall time complexity of the resolution scheme in a distributed environment is bounded to a constant factor of the resolution Algorithm 3.

5. Conclusion

In this paper, we formalized conflict detection and resolution in a context-aware authorization system. A typical context-aware authorization policy model is presented leveraging formalized context constraints. Specificity relations concerning contextual information are discussed and formally defined. Then, a novel graph-based approach is proposed to enable precedence establishment among authorizations in a conflict situation. The method is capable of using expressive resolution policies based on context and considers all authorization in a conflict situation as a whole. In the detection phase, a potential conflict graph is constructed, which is almost statically performable. Leveraging this graph in the actual conflict situation provides cost-effective context-based conflict resolution. In addition, timing strategy and detailed algorithms are provided and analyzed.

References

[1] M. A. Al-Kahtani and R. S. Sandhu. Induced role hierarchies with attribute-based rbac. In *8th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 142–148, Como, Italy, 2003. ACM.

[2] J. Al-Muhtadi, A. Ranganathan, R. H. Campbell, and M. D. Mickunas. Cerberus: A context-aware security scheme for smart spaces. In *1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 489–496, Fort Worth, Texas, USA, 2003. IEEE Computer Society.

[3] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[4] N. Dunlop, J. Indulska, and K. Raymond. Methods for conflict resolution in policy-based management systems. In *7th IEEE International Enterprise Distributed Object Computing Conference*, pages 98–109, Brisbane, Australia, 2003. IEEE Computer Society.

[5] W. Han, J. Zhang, and X. Yao. Context-sensitive access control model and implementation. In *5th International Conference on Computer and Information Technology (CIT 2005)*, pages 757–763, Shanghai, China, 2005. IEEE Computer Society.

[6] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, CA, USA, 1997. IEEE Computer Society.

[7] E. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, 1999.

[8] A. R. Masoumzadeh, M. Amini, and R. Jalili. Context-aware provisional access control. In *2nd International Conference on Information Systems Security*, volume 4332 of *Lecture Notes in Computer Science*, pages 132–146, Kolkata, India, 2006. Springer Verlag.

[9] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.

[10] C. Ruan and V. Varadharajan. A formal graph based framework for supporting authorization delegations and conflict resolutions. *International Journal of Information Security*, 1(4):211–222, 2003.

[11] P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design (FOSAD)*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer-Verlag, 2001.

[12] R. S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[13] E. Syukur, S. W. Loke, and P. Stanski. Methods for policy conflict detection and resolution in pervasive computing environments, May 10-14 2005.

[14] R. K. Thomas and R. S. Sandhu. Models, protocols, and architectures for secure pervasive computing: Challenges and research directions. In *2nd IEEE Conference on Pervasive Computing and Communications Workshops*, pages 164–170, Orlando, FL, USA, 2004.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE