ARGUMENT-BASED APPROACHES IN PRIORITIZED CONFLICTING SECURITY POLICIES

Salem Benferhat and Rania El Baida
CRIL-Université d’Artois
Rue Jean Souvraz, SP 18, 62307 Lens

Keywords: Organization-based access control, prioritized security policies, handling conflicts, argumentation reasoning.

Abstract: Information security system is an important problem in many domains. Therefore, it is very important to define security policies to restrict access to pieces of information in order to guarantee security properties, i.e. confidentiality, integrity and availability requirements. The joint handling of confidentiality, integrity and availability properties raises the problem of potential conflicts. The objective of this paper is to propose tools, based on the argumentation reasoning, for handling conflicts in prioritized security policies.

1 INTRODUCTION

Since information systems are more and more frequently used to manage sensitive and critical data, information security has become a major challenge. In this paper, we assume that a given information system manages both sensitive or critical data and the objective is to control access to these data in accordance with a security policy. This security policy will typically specify access control rules to protect data from unauthorized reading (confidentiality requirement) and unauthorized modification (integrity requirement). Another security requirement is to guarantee that data are accessible and usable upon demand by an authorized user (availability requirement). Ensuring the security of an information system comes down to check that these three requirements are satisfied.

Several access control systems have been proposed (Georgiadis et al., 2001; Sandhu et al., 1996; Wilikens et al., 2002). This paper deals with an access control model, called OrBAC (Organization-Based Access Control) (Abou El Kalam et al., 2003). In practice, knowledge bases, specifying security rules, are sometimes conflicting. For instance, a conflict may happen when a security policy defines an action that a user is permitted to perform and there exist some exceptional situations where performing such action is not acceptable.

There have been several proposals for handling conflicts in propositional knowledge bases (Dung, 1993; Benferhat et al., 1995; Besnard and Hunter, 2001; Brewka, 1989). Some of these approaches, called "argument-based approaches", accept inconsistency and cope with it. They retain all available information and suggest to select one or several consistent subbases which support or reject a conclusion. We show that contrary to the propositional case, some of argument-based approaches are not appropriate when dealing with conflicting first-order knowledge bases. We illustrate this statement with the safely-supported approach, and show that it suffers from the so-called "drowning problem" (formulas outside of conflicts are not recovered). We propose a solution to handling conflicts in first order knowledge bases.

The rest of this paper is organized as follows. Section 2 presents basic concepts of OrBAC system. Section 3 gives a prioritized first-order logic framework formalization of the OrBAC system. In section 4, two argumentation-based methods for handling conflicts in the OrBAC system are investigated.

2 ORGANIZATION-BASED ACCESS CONTROL (OrBAC)

The main features of OrBAC systems is that privileges are not directly assigned to users, actions and objects but to their respective abstractions called roles, activities and views. Moreover, OrBAC allows to represent security policies that depend on organizations and contexts. Following subsections give main concepts of the OrBAC system illustrated by Figure 1 (for more details see (Abou El Kalam et al., 2003)).
Basic Concepts  We distinguish two types of basic concepts: concrete basic concepts and abstract basic concepts. The basic concrete concepts are organizations, subjects, objects and actions. An Organization can be seen as an organized group of subjects playing some roles. A Subject is basically a user. The entity Organization covers inactive entities. The entity Action contains computer actions.

Basic abstract concepts are represented by the entities Role, View and Activity. The entity Role is associated with subjects that fulfill same functions. The entity View corresponds to a set of objects that satisfy common properties. The entity Activity corresponds to actions that share same principles.

Roles are organized in hierarchies. For instance, physicians are staffs, which means that if an organization employs a user in a role of physician, then the user is also employed in the role of staff.

Hierarchies between roles can be represented by the relationship Subrole(organization, role1, role2), which means that within the organization, role1 is a subrole of role2. In a similar way, we suppose that views and activities are organized in form of hierarchies.

Concrete concepts are related to abstract concepts by using the relationships Employ, Use and Consider. The relationship Employ(organization, subject, role) means that the organization employs a subject in a role. Similarly, the relationship Use(organization, object, view) means that the organization uses the object in a view. And the relationship Consider(organization, role, action) means that the organization considers that action falls within the activity.

Representing Abstracts Privileges and Contexts  A security policy is a set of permissions and prohibitions which are not directly defined on users, objects and actions but on their abstraction roles, views and activities. In OrBAC model, they are defined using the relationships Permission and Prohibition. The relationship Permission(organization, role, action, view, context) means that the organization grants to a role a permission to perform an action on a view within the context. The relationship Prohibition(organization, role, action, view, context) is defined similarly.

Contexts are used to specify the circumstances where organizations grant roles privileges to perform actions on views. Contexts are defined using the relationship Define. The relationship Define(organization, subject, action, object, context) means that within the organization, context is true between subject, object and action.

Concrete Privileges  Last concepts in OrBAC concern concrete actions that may be performed by subjects on concrete objects. For this purpose, the relationships Is-permitted and Is-prohibited are introduced. The relationship Is-permitted(subject, action, object) means that the subject is permitted to perform the action on the object. Is-prohibited is defined in a similar way. As we will see in the next section, these relationships are generally logically derived from abstract permissions and prohibitions granted to roles, views and activities.

3 PRIORITIZED LOGICAL-BASED ENCODING OF OrBAC SYSTEM

In this section, we propose a prioritized first-order logic encoding of OrBAC system. Prioritized first-order logic are simple extensions of first-order logic, by associating each classical first-order formula a level of priority. More precisely, a set of prioritized formulas is a set of weighted formulas having the form \( \{ (\phi_i, a_i), i = 1, n \} \), where \( \phi_i \) is a first-order formula and, \( a_i \) belongs to \([0, 1]\). The degrees \( a_i \)'s can simply express a preference relation between different formulas of the knowledge base. In this case, prioritized formulas can be put in a stratified form represented by \( \Sigma = S_1 \cup \cdots \cup S_n \), where each \( S_i \) contains classical first-order logic formulas. \( S_1 \) contains the most priority formulas and, \( S_n \) contains the least ones. And more generally, formulas in \( S_i \) have a same priority level and are more preferred than those of \( S_{i+1} \).

Prioritized first-order logic is enough for encoding OrBAC system, since deontic modalities (Permission and Prohibition) only bear on elementary actions (there is no disjunction of deontic modalities, no nested modalities ...).

Given an OrBAC model, given in entity-relationship format, we can construct its associated first-order knowledge base as usual. Namely, OrBAC knowledge bases are built over a language where the constant symbols correspond to the instances of the entities of the OrBAC diagram. The constant symbols and the individual variables \((x,y,z,\ldots)\) can be of types: Organization, Subject, Object, Action, Role, View, Activity and Context, which represent the domains of the OrBAC entities. The function symbols \((f(x,y), g(x,y,z), \ldots)\) are used for describing the entities attributes. The terms of the language consist of variables, constants and functions applied to these variables or constants. The relations symbols (predicates) correspond to the relationships of the OrBAC diagram (Figure 1). We use binary relations to compare the values of the entities attributes (e.g. \(=, >, <,\ldots\)). The following gives more precision on the structure and stratification of OrBAC knowledge bases.
Different Components Of OrBAC Knowledge Base

OrBAC knowledge base has three components (see Figure 2). The first one is the set of constraints that should be absolutely satisfied. The second part is composed of three parts: set of facts and formulas associated with the OrBAC relationships, set of facts and formulas associated with the OrBAC entities attributes and, set of inheritance rules. The last part of the knowledge base is the set of rules that allow to jump from abstract privileges to concrete privileges.

Formulas encoding constraints: We distinguish two kinds of constraints: constraints relating to the mutual exclusion and deontic modalities constraints. The mutual exclusion constraints concern the separation of roles, views, activities and contexts. An example of constraints of separation of roles can be: Within the organization A, a same user should not be assigned to role “physician” and role “nurse”, which is modeled by: \( \forall s, \neg \text{Employ}(A, s, \text{physician}) \lor \neg \text{Employ}(A, s, \text{nurse}) \).

The second kind of constraints relates to the link between concrete modalities. The following axiom links concrete permissions Is-permitted to concrete prohibitions Is-prohibited: \( \forall s, \forall o, \neg \text{Is-permitted}(s, o, a, \alpha, c) \land \neg \text{Is-prohibited}(s, o, a, o) \).

Formulas of OrBAC relationships, entities, inheritance rules: OrBAC relationships are encoded by first-order predicates. They can be given as facts. For instance, we can have:

- \( \text{Employ}(A, \text{Mary}, \text{secretary}) \).
- \( \text{Permission}(A, \text{staff}, \text{write}, \text{adm-rec}, \text{default}) \).
- \( \text{Subrole}(A, \text{physician}, \text{staff}) \).

They can also be provided and completed by first-order formulas. For instance, a rule of the form “if one is permitted to write JO’s administration record then he is permitted to read this object” can be written as: \( \forall s, \text{Is-permitted}(s, \text{write}, \text{rec-JO}) \rightarrow \text{Is-permitted}(s, \text{read}, \text{rec-JO}) \).

Roles’ hierarchies are described by formulas which relate on Employ’s relationship: \( \forall s, r_1, r_2, \text{Employ}(org, s, r_1) \land \text{Subrole}(org, r_1, r_2) \rightarrow \text{Employ}(org, s, r_2) \), means that if a subject \( s \) is employed in the role \( r_1 \) and if \( r_1 \) is a subrole of \( r_2 \), then \( s \) is also employed in role \( r_2 \).

Hierarchies between activities and views are defined in a similar way.

Set of jumping rules: The last part of the OrBAC knowledge base is a set of rules which allow to jump from abstract privileges into concrete privileges. This transformation is given by the following axiom (for each organization, role, view, activity and context explicitly stated in the knowledge base):

\[ \forall s, \forall a, \forall o, \forall \alpha, \forall \alpha', \text{Consider}(org, \alpha, a) \land \text{Define}(org, s, \alpha, o, c) \land \text{Is-permitted}(s, \alpha, o) \land \neg \text{Is-prohibited}(s, \alpha, o) \rightarrow \text{Is-permitted}(s, a, o) \land \text{Is-prohibited}(s, a, o) \]

If the organization \( org \), within the context \( c \), grants role \( r \) permission to perform activity \( a \) on view \( v \), and if \( org \) employs subject \( s \) in role \( r \), and if \( org \) uses object \( o \) in the view \( v \), and if \( org \) considers that action \( \alpha \) falls within the activity \( a \) and if, within the organization \( org \), the context \( c \) is true then \( s \) is permitted to perform \( \alpha \) on \( o \).

The transformation from abstract prohibition to concrete prohibition is defined in a similar way. These jumping rules have not the same level of priorities and are stratified, since there are situations in which they should not be applied.

The stratification of these rules can be given by an expert or automatically computed. It also depends on...
conflicts resolution strategies. A strategy of resolution of conflicts can be for example that rules encoding "prohibitions" are always preferred to rules encoding "permissions". However, this is not satisfactory since it can occur that certain specific rules are ignored and never be used. In the OrBAC system, conflicts are generally due to exceptions of the privileges inheritance induced by the hierarchy between roles, views and activities. In this paper, we propose to use the algorithm of "Minimum Specificity Principle" (MSP) detailed in (Benferhat et al., 1997). The idea in this algorithm is first to consider that rules without exceptions and facts are always preferred to rules with exceptions. Then, a rule encoding an exceptional situation is preferred to a rule encoding a general situation. A rule is considered as "exceptional" if letting its antecedent to be true leads to an inconsistency.

4 ARGUMENT-BASED APPROACHES FOR HANDLING CONFLICTS IN OrBAC SYSTEM

Several approaches have been proposed for handling conflicts in propositional knowledge bases (Benferhat et al., 1995; Besnard and Hunter, 2001; Brewka, 1989; Dung, 1993). Some of these approaches, called "argument-based approaches", accept inconsistency and cope with it. They retain all available information and suggest to select one or several consistent subbases which support or reject a conclusion. In this section, we investigate two notions of consequence based on the argument-based reasoning. The first one provides, for each access control request, a best argument that supports a permission or a prohibition of access. The second one distinguishes between safe and unsafe arguments.

4.1 Argued Consequence

A conclusion can be inferred from an inconsistent knowledge base if the latter contains an argument that supports this conclusion with some priority level, but there is no argument that supports its negation with a higher or equal level of priority. More formally, the argued consequence is summarized by the two following definitions:

**Definition 1** A subbase $A$ of $\Sigma$ is said to be an argument of rank $i$ for a formula $\phi$, if it satisfies the following conditions:

1. $A \not\vdash \bot$ (consistency),
2. $A \vdash \phi$ (relevance), and
3. $\forall \psi \in A, A - \{\psi\} \not\vdash \phi$ (economy).
4. $R(A) = \max\{j : A \cap S_j \neq \emptyset\} = i$.

Namely, an argument in favour of a conclusion $\phi$ is a smallest consistent subbase of $\Sigma$ that infers $\phi$. $R(A)$ represents the degree of support of $\phi$ by $A$; it corresponds to the rank of the least priority formula in $A$.

**Definition 2** A formula $\phi$ is said to be an argued consequence of $\Sigma$, denoted by $\Sigma \vdash A \phi$, if and only if:

1. there exists an argument of rank $i$ for $\phi$ in $\Sigma$, and
2. all arguments for $\neg \phi$ (if any) are of rank $j > i$.

**Example 1** Let us consider an organization "Hospital A", where we have one patient ("JO") and two staffs: one physician ("Bob") and one secretary ("Mary"). Assume that we have one object: JO’s administration record. We assume that the security policy of this organization is represented by the following rules:

1. A staff member has a permission to write patients’ administration records.
2. A physician has a prohibition to write patients’ administration records.
3. A physician is a staff member.
4. A secretary is a staff member.
5. Regarding JO's administration record, if one is permitted to write this record then he is permitted to read this record.

Let us suppose that security policy rules lead to the following prioritized knowledge base
\[ \Sigma = S_1 \cup S_2 \cup S_3 \cup S_4, \]
with
\[ S_1 = \{ R_1, \forall s, \alpha, o, \neg Is\_permitted(s, \alpha, o) \vee \neg Is\_prohibited(s, \alpha, o) \}; \]
\[ S_2 = \{ R_3, Subrole(A, physician, staff) \}; \]
\[ R_4, \text{Employ}(A, Bob, physician); \]
\[ R_5, \text{Employ}(A, Mary, secretary); \]
\[ R_6, \text{Consider}(A, read, consult); \]
\[ R_7, \text{Consider}(A, write, write); \]
\[ R_8, Use(A, rec\_JO, adm\_rec); \]
\[ R_9, \text{Permission}(A, staff, write, adm\_rec, default); \]
\[ R_{10}, \text{Prohibition}(A, physician, write, adm\_rec, default); \]
\[ R_{11}, \forall s, Is\_permitted(s, write, rec\_JO) \rightarrow Is\_permitted(s, read, rec\_JO); \]
\[ R_{12}, \forall s, \text{Employ}(A, s, physician) \wedge Subrole(A, physician, staff) \rightarrow \text{Employ}(A, s, staff); \]
\[ R_{13}, \forall s, \text{Employ}(A, s, secretary) \wedge Subrole(A, secretary, staff) \rightarrow \text{Employ}(A, s, staff); \]
\[ R_{14}, \forall org, s, \alpha, o, \text{Define}(org, s, \alpha, o, default) \equiv \top \}; \]
\[ S_3 = \{ R_{15}, \forall s \forall \alpha \forall o, \]
\[ \text{Prohibition}(A, \text{physician}, write, \text{adm\_rec}, \text{default}) \wedge \text{Employ}(A, s, \text{physician}) \wedge \text{Use}(A, o, \text{adm\_rec}) \wedge \text{Consider}(A, \alpha, write) \wedge \text{Define}(A, s, \alpha, o, default) \rightarrow \text{Is\_prohibited}(s, \alpha, o) \}. \]
\[ S_4 = \{ R_{16}, \forall s \forall \alpha \forall o, \]
\[ \text{Permission}(A, staff, write, \text{adm\_rec}, \text{default}) \wedge \text{Employ}(A, s, staff) \wedge \text{Use}(A, o, \text{adm\_rec}) \wedge \text{Consider}(A, \alpha, write) \wedge \text{Define}(A, s, \alpha, o, default) \rightarrow \text{Is\_permitted}(s, \alpha, o) \}. \]

We are interested to know if Bob is permitted to write JO's administration record.

We have an argument
\[ B = \{ R_4, R_7, R_{10}, R_{11}, R_{12}, R_{14}, R_{15} \}, \]
of rank 3 for Is-prohibited(Bob, write, rec-JO).

There is only one argument
\[ C = \{ R_2, R_4, R_7, R_{10}, R_{11}, R_{12}, R_{14}, R_{15} \} \]
for Is-permitted(Bob, write, rec-JO) which is of rank 4.

The following proposition shows that applying argued-consequence relation on a first-order knowledge base or on its instantiated base is equivalent. We are restricted to a finite domain, namely to the case when the instantiated base is finite.

**Proposition 1** Let \( \Sigma = S_1 \cup \ldots \cup S_n \) be an inconsistent first-order knowledge base, and let \( \psi \) be an instantiated formula. Let \( \text{Instantiate}(\Sigma) \) be the base obtained by instantiating all first order formulas in \( \Sigma \) with only constant symbols appearing in \( \Sigma \) (we assume that there exist at least one constant symbol before instantiating). If \( \text{Instantiate}(\Sigma) \) is finite, then \( \Sigma \vdash_A \psi \iff \text{Instantiate}(\Sigma) \vdash_A \psi \).

### 4.2 Safely Supported Consequence

The second approach that we investigate, called "safely supported inference", can be summarized by the two following definitions.

**Definition 3** A formula \( (\phi, i) \) is said attacked if there exists an argument of rank \( j \) for \( \neg \phi \) such that \( j < i \). A subbase \( A \) is a safe argument if \( \exists \phi \in A \) such that \( \phi \) is attacked.

**Definition 4** A formula \( \psi \) is said to be a safely supported consequence of \( \Sigma \), denoted by \( \Sigma \vdash_{SS} \psi \), iff there exists in \( \Sigma \) a safe argument for \( \psi \).

The safely supported relation is satisfactory in propositional bases. However, it fails in first-order knowledge bases, since it suffers from the drowning problem (formulas outside of conflicts are not recovered) as it is illustrated by the following example.

**Example 2** Let us take again example 1. There exists an argument of rank 4 for Is-permitted(Mary, read, rec-JO);
\[ B = \{ R_3, R_5, R_7, R_8, R_9, R_{11}, R_{13}, R_{14}, R_{16} \}. \]
However, \( B \) is not a safe argument. Indeed, the formula \( (R_{16}, A) \) is attacked by the argument \( C = \{ R_4, R_7, R_8, R_{10}, R_{12}, R_{14}, R_{15} \} \) of rank 3. Hence, Is-permitted(Mary, read, rec-JO) is not a consequence of \( \Sigma \).

This limitation is explained by the fact that the formula "R_{16}" is attacked (for instance Bob) and thus, will be drawn aside by the safely supported relation. By drawing aside this formula, it cannot be applied for Mary which leads to the drowning problem.

### 4.3 Adaptation of Safely Supported Consequence

The limitation of the safely supported consequence are explained by the fact that drawing aside a first-order formulas comes back to drawing aside a set of propositional formulas (which are not necessarily tackled). This is not satisfactory because if a formula
is responsible for a conflict, then it is not the case that all its instances are also responsible for the conflict. We distinguish two ways in which a formula can be attacked:

- Weakly attacked: if only some instances of this formula are attacked.
- Strongly attacked: if all instances are attacked.

We propose to redefine the safely supported relation for first-order knowledge bases. The idea is that a conclusion can be inferred from an inconsistent knowledge base if the latter contains an argument that supports this conclusion such that there is no strongly attacked formula in this argument. The counterpart of the safely supported inference is as follows:

**Definition 5** A formula \( \langle \phi(x), i \rangle \) is weakly attacked if, there exists an instance \( x_k \) such that, there exists an argument of rank \( j \) for \( \neg \phi(x_k) \) such that \( j < i \). A formula \( \langle \phi(x), i \rangle \) is strongly attacked if, \( \forall x \), there exists an argument of rank \( j \) for \( \neg \phi(x) \) such that \( j < i \).

**Definition 6** A formula \( \psi \) is said to be a strongly consequence of \( \Sigma \), denoted by \( \Sigma \vdash_S \psi \), iff there exists an argument \( A \) for \( \psi \) such that \( \exists \phi \in A, \phi \) is weakly attacked.

A formula \( \psi \) is said to be a weakly consequence of \( \Sigma \), denoted by \( \Sigma \vdash_W \psi \), iff there exists an argument \( A \) for \( \psi \) such that \( \exists \phi \in A, \phi \) is strongly attacked.

**Example 3** Let us consider example 2. The subbase \( B \) is the only argument for \( \text{Is-permitted}(\text{Mary, read, rec-JO}) \) of rank 4. However, \( B \) contains the formula \( R_{16} \) which is weakly attacked by the argument \( C \) of rank 3, for instance \( (s = \text{Bob}, \alpha = \text{read and, } o = \text{rec-JO}) \). Hence, \( \text{Is-permitted}(\text{Mary, read, rec-JO}) \) is not a strongly consequence of \( \Sigma \), i.e. \( \Sigma \not\vdash_S \text{Is-permitted}(\text{Mary, read, rec-JO}) \). \( B \) does not contain any strongly attacked formula, thus \( \text{Is-permitted}(\text{Mary, read, rec-JO}) \) is a weakly consequence of \( \Sigma \), i.e. \( \Sigma \vdash_W \text{Is-permitted}(\text{Mary, read, rec-JO}) \).

## 5 CONCLUSION

This paper proposed to equip the flexible access control system OrBAC with a conflict resolution module. Two approaches, based on the argumentation reasoning, have been proposed. The argued consequence is very intuitive. Indeed, it retains all available information and suggests to select one or several arguments which support or reject a permission or a prohibition of access. However, this method is not entirely satisfactory. Indeed, it can lead to undesirable conclusions. This limitation is explained by the fact that one argument may for instance contain pieces of information which are directly involved in the inconsistency of the knowledge base. The safely supported consequence only delivers safe conclusions. However, this method is not appropriate when dealing with inconsistent first-order knowledge bases. We showed how to rephrase the safely supported relation in the framework of first-order logic.

## ACKNOWLEDGMENTS

This work is supported by the national ACI project DESIRS.

## REFERENCES


