Conflict Handling Strategies for Partially Ordered Access Control Security Policies

Ahmed Laouar, Sihem Belabbes, Salem Benferhat

Conflict Handling Strategies for Partially Ordered Access Control Security Policies

Ahmed Laouar
CRIL, Univ. Artois & CNRS
UMR 8188, Lens, France
laouar@cril.fr

Sihem Belabbes
LIASD, IUT de Montreuil
Univ. Paris 8, Saint-Denis, France
belabbes@iut.univ-paris8.fr

Salem Benferhat
CRIL, Univ. Artois & CNRS
UMR 8188, Lens, France
benferhat@cril.fr

***Abstract*—In access control security models, the users of an organization's information system may be granted with conflicting privileges. This is usually the case when the underlying security policy implements both permission and prohibition rules. In this paper, we propose to capture uncertainty in security models, within the framework of possibility theory. We define efficient strategies for handling conflicting privileges derived by a security policy, based on priorities assigned to the permissions and prohibitions. We show that these strategies are in line with the possibilistic management of inconsistency in security policies.**

***Index Terms*—Access Control Security Models, Inconsistency Management, Prioritized OrBAC.**

I. Introduction

Data protection and data security are major concerns for organisations and companies dealing with large-scale information systems. Furthermore, the collection and processing of sensitive data must comply with strict regulations. Thus, it is crucial to control access to an organization's IT resources in order to safeguard user confidentiality and data integrity.

Access control security models offer a formal specification for security policies, which are a set of rules that regulate access to the resources of a system. They allow the expression of authorization rules and conditions to determine whether a user (subject) has the permission to perform an action on a given resource (or object). Different access control models have been proposed in order to enhance the scope of application and use of security policies in a formal and standardized way (see [16], and [21] for a non-exhaustive list).

The main question here is what happens if both a grant and a deny decision are returned following an access request. This situation arises from the simultaneous use of different types of privilege rules, such as permissions and prohibitions. In fact, the simpler models address this issue by avoiding the storage of conflicting rules. Unfortunately, this approach is not scalable, as sophisticated and dynamic models derive privileges from prior information on the policy elements.

Among expressive access control models, the Organisation-Based Access Control model (OrBAC) [18] is a dynamic and contextual model, that allows to express several types of authorisation rules, including permission, prohibition, obligation and recommendation. The presence of these rules together may lead to authorization conflicts. In that case, the model is described as inconsistent. This highlights the need for appropriate conflict handling strategies.

Furthermore, the OrBAC model supports a single policy per organization, established by a central entity. However, it does not discuss the applicability of multiple policies at the same time. A similar situation can, on top of driving to conflicts, lead to the need of making a decision from incomparable policies, i.e., no preference relation holds between them. So, while the single policy/authority allow a natural use of total orders, having multiple policies may imply the need to employ partial orders. Partial orders can also result from applying different preference relations to various elements of the policy, such as using distinct preferences for contexts and roles.

A large body of work has been devoted to the management of conflicting information in areas such as propositional logic, databases and description logics [3], [9]. When faced with a conflict, one can adopt either a risky approach, which involves modifying or rewriting a portion of the conflicting data, or a cautious approach that allows for reasoning in the presence of conflicts. In the latter, one can employ a preference strategy to discriminate a part of the conflict. Alternatively, a repair-based strategy can be used, wherein an inconsistent knowledge base is replaced with one or more of its consistent sub-bases.

This paper aims to set an automated approach for handling conflicts in dynamic access control models, and reasoning with incomparable policies. The main contributions are:

- We adapt the notion of prioritized OrBAC by introducing fully certain and uncertain rules.
- We address the problem of conflicts handling in partially ordered OrBAC using a direct query-oriented method.
- We propose a new method, called the OrBAC repair, inspired by inconsistency-tolerant semantics. The method is extended efficiently to partially ordered OrBAC.

This paper is organized as follows, Section II discusses related work. Section III recalls the OrBAC model. Section IV introduces the notions of priorities and inconsistency in OrBAC, alongside a query-oriented method to resolve the conflicts. Section V introduces an efficient method to resolve the conflicts. We illustrate the proposed methods with an example in Section V-C, before concluding with a discussion.

II. Related work

The problem of handling conflicts in access control have been widely addressed in the literature, under multiple frameworks, the reader may refer to [17] for an overview of the available and the possible conflict detection and resolution methods (as part of policy analysis methods). Some of the discussed methods adopt a human assisted approach (as in [12]), where some conflicting queries are still posed to an administrator. Some other methods are based on argumentation and deal with policies as arguments and possible conflicts as attacks in the argumentation framework [10].

Conflict handling in OrBAC has already been investigated. However, the existing methods either consider a particular cause for the emergence of conflicts (e.g. the presence of privileges rules with exceptions [8]), or use a general encoding with first-order logic [7], resulting in computational challenges. Other methods (e.g. [6]) propose to introduce the notion of possibilistic priorities into the OrBAC model without addressing the issue of conflicting information.

Moreover, the authors in [15] proposed a formal way to handle conflicts in OrBAC. Their approach addresses the shortcomings of handling conflicts in Rule-Based models. Mainly by detecting potential conflicts between the abstract elements of a policy to avoid conflicts on the concrete level. The method prevents the evolution of the model in a direction that leads to a conflict. It uses a prioritized version of OrBAC to select a more preferable rule, when a total order relation applies between the conflicting rules. When rules are incomparable (a partial order is defined on the model), it requires the definition of separation constraints between the abstract elements to avoid conflicts. The authors demonstrate that checking a potential conflict in the prioritized OrBAC is done in polynomial time.

Research efforts in inconsistency-tolerant semantics have explored the notion of repair, however, repair-based approaches are hampered by high computational complexity, except in particular frameworks such as those based on lightweight ontologies. Indeed, the DL-Lite family of description logics [13] enjoy favourable computational properties and a reasonable expressive power, which make them a formalism of choice for many applications. For instance, DL-Lite has been used to deal efficiently with conflicting information in data access and query answering [3], [9].

III. OrBAC

Since its first proposition, OrBAC [18] attracted a lot of interest in the specification and modelling of security policies. Thanks to its ability to express a large set of access control rules, while retaining a compact and abstract definition of the policy. In addition, concrete access rules are inferred in OrBAC. In the literature, several recent applications of the OrBAC model exist, notably in domains like Security Protocols [2], IoT [19] and Safety Engineering [14]. Moreover, the model has been extended in different ways [1], [11].

The elements of a policy in OrBAC are divided into concrete and abstract concepts. The concrete concepts are subject, object and action. They represent respectively a user (a process or a human), an inactive entity (like a shared resource or a file) and an access operation (like read and write). Moreover, OrBAC also considers organizations to be concrete concepts. Abstract concepts provide a means of generalizing (or abstracting) concrete concepts. Hence, subjects that fulfill the same functions are grouped in the same role, objects that satisfy common properties are grouped in the same view and actions that share the same principles are grouped into an activity. A subject can play multiple roles, the same applies for objects and actions.

OrBAC links the concrete elements of a policy with the abstract elements using the rules Employ, Use, and Consider. Employ(org,s,r) means that the subject "s" plays the role "r" within the organization "org". Use(org,o,v) means that the object "o" is used in the view "v" within the organization "org". Consider(org,α,a) means that the organization "org" considers that the action "α" fall within the activity "a". Furthermore, a context in OrBAC determines the circumstances under which an organization grants or denies access privileges. Indeed, a subject may have the permission to access an object in one context and lose that same permission in another context. Contexts are defined using the rule Define(org,s,α,o,c), which is interpreted as: in the organization "org" the context "c" is true between the subject "s", the object "o" and the action "α".

The security policy is defined on roles, views and activities via the rules: Permission, Prohibition, Obligation, and Recommendation. Each of these rules has five arguments. The rule Permission(org, r, a, v, c) means that the organization "org" grants the role "r" the permission to perform the activity "a" on the view "v" if the context "c" holds. The rules Prohibition, Obligation, and Recommendation can be defined in the same way.

Concrete privileges represent the derived access rights of subjects on objects with actions. These privileges are Is-permitted, Is-prohibited, Is-obliged, and Is-recommended. The privilege Is-permitted(s,α,o) means that the subject "s" is allowed to perform the action "α" on the object "o". The other privileges can be defined in the same way. The derivation of these concrete privileges is given by the following definition:

Definition 1 (Derivation of concrete rules)**.** *Concrete permissions are derived from abstract permissions using the following rule:*

$\forall$org, $\forall$s, $\forall\alpha$, $\forall$o, $\forall$r, $\forall$a, $\forall$v, $\forall$c,

Permission(org,r,a,v,c) $\wedge$ Employ(org,s,r) $\wedge$ Use(org,o,v) $\wedge$ Consider(org, α, a) $\wedge$ Define(org,s,α,o,c) $\models$ Is-permitted(s,α,o)

The other types of privileges are derived in the same way.

An instance of OrBAC is a knowledge base containing the abstract and the concrete elements of the policy, the rules connecting them and the set of the abstract rules.

Definition 2 (OrBAC instance)**.** *An instance of OrBAC is a tuple of the form* $\mathcal{O} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P} \rangle$*. Such that* $\mathcal{B}$ *is a set of basic concrete and abstract concepts,* $\mathcal{U}$ *is a set of rules of the form* Employ, Use, Consider *and* Define *and* $\mathcal{P}$ *is the set of the abstract rules* Permission, Prohibition, Obligation *and* Recommendation.

In the remaining of this paper, we denote by $\mathcal{O} \models$ Is-permitted$(\mathsf{s}, \alpha, \mathsf{o})$ (resp.$\mathcal{O} \not\models$ Is-permitted$(\mathsf{s}, \alpha, \mathsf{o})$) the fact that

a concrete permission is derived (resp. is not derived) from the instance $\mathcal{O}$ (the same applies for other types of privileges).

IV. Conflict handling in OrBAC

In this section, we present a new representation of priorities in OrBAC. This allows us to establish the notion of conflicts, before introducing methods to resolve them.

A. Prioritized OrBAC

A crucial difference between the proposed representation and those found in the literature [7], [8], [15], is the separation of the policy elements into fully certain and uncertain rules. This has already been discussed in [7], however, the proposed separation does not apply uniformly throughout the model. Our representation draws a parallel with lightweight ontologies, where separation is made between generic and factual elements in knowledge bases.

The fully certain elements are the abstract privileges of the policy, which are the set $\mathcal{P}$ in an OrBAC instance. The uncertain elements are the rules mapping concrete concepts to abstract concepts in the policy, which form the set $\mathcal{U}$ in an OrBAC instance. The uncertain elements are associated with priorities from a non-empty ordered set $\mathcal{L} = \{w_1, \ldots, w_n, \mathbb{1}\}$. The order defined over the elements of $\mathcal{L}$ is an irreflexive and transitive binary relation which can either be:

- a total order relation, denoted $>$ such that $w_i > w_{i-1}$ (for $i = 2, \ldots, n$), or
- a partial order relation, denoted $\rhd$ such that some elements of $\mathcal{L}$ may be incomparable, i.e., none of $w_i \rhd w_{i+1}$ or $w_{i+1} \rhd w_i$ holds. Incomparability is denoted by $\bowtie$.

Note that in both cases, the priority $\mathbb{1}$ represents full certainty: $\forall i,\ \mathbb{1} > w_i$ (resp. $\mathbb{1} \rhd w_i$).

Definition 3 (Prioritized OrBAC)**.** *A prioritized OrBAC instance is a tuple $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ such that $\mathcal{B}$, $\mathcal{U}$ and $\mathcal{P}$ are OrBAC instance elements as given by Definition 2, and $\mathcal{L} = \{w_1, \ldots, w_n, \mathbb{1}\}$ is a non-empty ordered set of priorities associated with the elements of $\mathcal{U}$. Namely,* Employ(org,s,r,w_i), Use(org,o,v,w_j), Consider(org,α,a,w_k) and Define(org,s,α,o,c,w_h), where org, s, r, o, v, α, a, c $\in \mathcal{B}$.

This representation of uncertainty leads to some specific aspects of the notion of conflicts:

- Conflicts cannot occur between the abstract rules, which means that both a Permission and a Prohibition cannot be defined for the same abstract concepts.
- The necessity to tolerate potential conflicts as defined in [15]. A potential conflict occurs when a Permission and a Prohibition are defined on abstract elements that might generalize the same concrete elements.
- Conflict resolution is done on the concrete level, mainly, in the presence of an access query.

As a consequence, the model is less restrained and more suitable to dynamic environments. Moreover, it highlights the need for methods to act on conflicts when they occur rather than defining rules or conditions to avoid them.

The following definition adapts the derivation of concrete privileges to the case of a prioritized OrBAC instance:

Definition 4 (Prioritized derivation of concrete rules)**.** *Concrete permissions are derived from abstract permissions using the following rule:*
$\forall$org, $\forall$s, $\forall\alpha$, $\forall$o, $\forall$r, $\forall$a, $\forall$v, $\forall$c,
Permission(org,r,a,v,c) $\wedge$ Employ(org,s,r,w_1) $\wedge$ Use(org,o,v,w_2) $\wedge$ Consider(org,α,a,w_3) $\wedge$ Define(org,s,α,o,c,w_4) $\models$ Is-permitted(s,α,o,$\{w_1, w_2, w_3, w_4\}$).
The other types of privileges are derived in the same way.

The uncertainty over the resulting privilege is the combination of the uncertainty defined on its deriving elements. For the sake of simplicity, the uncertain rules of the form Employ(org,s,r,w_i), Use(org,o,v,w_i), Consider(org,α,a,w_i) and Define(org,s,α,o,c,w_i) are referred to using the notation $T(\cdot, w_i)$. In the remaining of this paper, we consider the following form of an access query: the subject s asks to perform the action α on the object o, namely, Is-permitted(s,α,o)?.

B. Inconsistency in OrBAC

The following definition introduces the notion of inconsistency in OrBAC:

Definition 5 (inconsistent OrBAC instance)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a prioritized OrBAC instance. $\mathcal{O}_\mathcal{L}$ is said to be inconsistent if there exists a subject "*s*", an object "*o*" and an action "α" such that both* Is-permitted(s,α,o,$\{w_1, w_2, w_3, w_4\}$) and Is-prohibited(s,α,o,$\{w_5, w_6, w_7, w_8\}$) are derived from $\mathcal{O}_\mathcal{L}$. Otherwise, $\mathcal{O}_\mathcal{L}$ is said to be consistent.

A conflict in OrBAC is the minimal set of rules used to drive both a concrete permission and a concrete prohibition for the same "s", "o" and "α". Formally:

Definition 6 (OrBAC conflict)**.** *Let $\mathcal{C}$ be a subset of uncertain and fully certain rules from a prioritized instance $\mathcal{O}_\mathcal{L}$. $\mathcal{C}$ is a conflict in $\mathcal{O}_\mathcal{L}$ if*

- $\mathcal{C} \models$ Is-permitted(s, α, o, $\{w_1, w_2, w_3, w_4\}$) *and* $\mathcal{C} \models$ Is-prohibited(s, α, o, $\{w_5, w_6, w_7, w_8\}$)*, and*
- $\forall T(\cdot, w_i)$ *an uncertain rule in $\mathcal{C}$,* $(\mathcal{C} \setminus \{T(\cdot, w_i)\}) \not\models$ Is-permitted(s, α, o, $\{w_1, w_2, w_3, w_4\}$) *or* $(\mathcal{C} \setminus \{T(\cdot, w_i)\}) \not\models$ Is-prohibited(s, α, o, $\{w_5, w_6, w_7, w_8\}$).

Note that unlike many other languages (like DL-Lite [13]) conflicts are not only binary, and may involve more elements. Moreover, in this paper, we consider only conflicts between permissions and prohibitions for the sake of simplicity; however, these methods can be easily extended.

A naive approach to compute all conflicts of an OrBAC instance is to derive all the concrete rules, then apply some search algorithm. This approach can be impractical when the number of rules is large. Instead, a simple conjunctive query can be executed on a relational database representation of the instance to get all the conflicts.

Proposition 1. *Computing all the conflicts of an OrBAC instance is a tractable task.*

Proposition 1 can be achieved using the following first order logic conjunctive query:

$$
\begin{aligned}
&Q_1(\mathsf{s}, \alpha, \mathsf{o}, \mathsf{r}_i, \mathsf{r}_j, \mathsf{a}_i, \mathsf{a}_j, \mathsf{v}_i, \quad \mathsf{v}_j, \mathsf{c}_i, \mathsf{c}_j, w_1, \ldots, w_8) = \\
&\mathsf{Permission}(\mathsf{org}, \mathsf{r}_i, \mathsf{a}_i, \mathsf{v}_i, \mathsf{c}_i) \quad \wedge \mathsf{Prohibition}(\mathsf{org}, \mathsf{r}_j, \mathsf{a}_j, \mathsf{v}_j, \mathsf{c}_j) \quad \wedge \\
&\mathsf{Employ}(\mathsf{org}, \mathsf{s}, \mathsf{r}_i, w_1) \quad \wedge \mathsf{Employ}(\mathsf{org}, \mathsf{s}, \mathsf{r}_j, w_5) \quad \wedge \\
&\mathsf{Use}(\mathsf{org}, \mathsf{o}, \mathsf{v}_i, w_2) \quad \wedge \mathsf{Use}(\mathsf{org}, \mathsf{o}, \mathsf{v}_j, w_6) \quad \wedge \\
&\mathsf{Consider}(\mathsf{org}, \alpha, \mathsf{a}_i, w_3) \quad \wedge \mathsf{Consider}(\mathsf{org}, \alpha, \mathsf{a}_j, w_7) \quad \wedge \\
&\mathsf{Define}(\mathsf{org}, \mathsf{s}, \alpha, \mathsf{o}, \mathsf{c}_i, w_4) \quad \wedge \mathsf{Define}(\mathsf{org}, \mathsf{s}, \alpha, \mathsf{o}, \mathsf{c}_j, w_8)
\end{aligned}
$$

C. The Query-Oriented Method

In the case of an inconsistent prioritized OrBAC instance, as given by Definition 5, a natural approach is to consider the more certain derived privilege related to the subject "s", the action "α" and the object "o". This means that a query of the form Is-permitted(s,α,o)? is granted if there exists a derived permission that is strictly more certain than any derived prohibition. This approach focuses only on the rules that are related to the elements "s", "α" and "o".

The following definition provides the conditions for a permission to be granted in the case of a total order relation ($>$).

Definition 7 (Query-oriented method)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a totally ordered OrBAC instance and $>$ be the total order defined over the priorities in $\mathcal{L}$. The query* Is-permitted(s,α,o)? is granted if:

1) $\mathcal{O}_\mathcal{L} \models$ Is-permitted(s,α,o,$\{w_1, w_2, w_3, w_4\}$) and $\mathcal{O}_\mathcal{L} \not\models$ Is-prohibited(s,α,o,$\{w_5, w_6, w_7, w_8\}$), or
2) $\exists w_1, w_2, w_3, w_4$ such that:
   - $\mathcal{O}_\mathcal{L} \models \mathsf{Is\text{-}permitted}(\mathsf{s}, \alpha, \mathsf{o}, \{w_1, w_2, w_3, w_4\})$, and
   - $\forall w_5, w_6, w_7, w_8$ s.t. $\mathcal{O}_\mathcal{L} \models \mathsf{Is\text{-}prohibited}(\mathsf{s}, \alpha, \mathsf{o}, \{w_5, w_6, w_7, w_8\})$:
     $$min\{w_1, w_2, w_3, w_4\} > min\{w_5, w_6, w_7, w_8\}$$

   (Where $min\{w_i, w_j, w_k, w_h\}$ is the least certain element).

Otherwise, the query Is-permitted(s,α,o)? is not granted.

In the case of a partial order ($\triangleright$), Definition 7 cannot be applied, as certain priorities in $\mathcal{L}$ are incomparable. One way to deal with this problem is to compute all the possible total order extensions of the partial order $\triangleright$. A family of total orders $(>_1, \ldots, >_m)$ is obtained such that: strict ordering is preserved: $\forall >_i$, if $w_j \triangleright w_k$ then $w_j >_i w_k$, and incomparability $w_j \bowtie w_k$ is extended to either $w_j >_i w_k$ or $w_k >_i w_j$.

Definition 8 (Partially ordered query-oriented method)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a partially ordered OrBAC instance and $\triangleright$ be the partial order defined over the priorities in $\mathcal{L}$. Let $(>_1, \ldots, >_m)$ be the total order extensions of $\triangleright$ (as defined above). The query* Is-permitted(s,α,o)? is granted if it follows from each total order extension $>_i$ associated with the instance $\mathcal{O}_\mathcal{L}$ using Definition 7.

Definition 8 ensures that a derived permission is not granted unless it is strictly preferred to any derived prohibition. However, checking if a query follows in this case using the query-oriented method of Definition 8 is not tractable if we explicitly use all the total order extensions of the partial order, since their number is exponential in the worst case.

V. The Repair-Based Method

In the above section, we showed that checking if a permission is granted by a partially ordered inconsistent OrBAC instance is a challenging task. In similar settings, tractable inconsistency-tolerant semantics have been introduced to deal with partially ordered lightweight ontologies [4], [5], [20].

In this section, we propose to apply some of these methods to resolve conflicts in OrBAC. This can be achieved in two ways. The first is encoding OrBAC in lightweight ontologies. However, this can either lead to a loss of expressiveness or to the use of a larger language, which means losing tractability. Moreover, the cost of mapping OrBAC instances to the used language must be taken into consideration. The second way, which we believe is more appropriate, consists in preserving the OrBAC model and applying the semantics on it.

A. OrBAC repair

The Cπ-repair method, which was introduced in [20], applies the possibilistic repair on each total order extension of the partial order defined on the uncertain elements of a knowledge base. The following definition adapts the notion of possibilistic repair to totally ordered OrBAC instances.

Definition 9 (OrBAC Repair)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a totally ordered OrBAC instance. Let $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L})$ be a subset of $\mathcal{O}_\mathcal{L}$ and w_i be the least certain priority associated with an element of $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L})$. $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L})$ is said to be a repair of $\mathcal{O}_\mathcal{L}$ if $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L})$ is consistent and $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L}) \cup \{T(\cdot, w_{i-1})\}$ is inconsistent (where $T(\cdot, w_{i-1}) \in \mathcal{U}$).*

Proposition 2. *If* Is-permitted(s,α,o) is derived from the repair $\mathcal{R}_\pi(\mathcal{O}_\mathcal{L})$ using the derivation rule given in Definition 4, then Is-permitted(s,α,o) is granted using Definition 7.

Now, when presented with a partially ordered OrBAC instance, we can follow the same logic of the Cπ-repair method to compute a repair:

1) First, compute all the total orders $(>_1, \ldots, >_m)$ that extend the partial order $\triangleright$.
2) Then, compute a repair for each total order extension $>_i$ associated with the instance, as per Definition 9.
3) Apply the derivation of Definition 4 on each repair of a total order extension, a set of all the concrete privileges associated with each repair is obtained.
4) Intersect all the sets of privileges to obtain a single repair.

The approach described above guarantees that a permission is granted if and only if it can be derived from every repair obtained from the total order extensions of the partial order.

B. Accepted permission

As we showed before, computing the total order extensions of a partial order is impractical. Instead, we propose a tractable and equivalent method that allows to check if a query follows from the described repair. The method is an adaptation of the characterization of the Cπ-repair to OrBAC instances. Moreover, the method is capable of checking if a permission is granted, without computing the repair or deriving all the

privileges of an instance. The method relies on three important notions. The conflicts, which can be computed using the query Q_1 from Proposition 1, and the concepts of support and dominance which are given by the following definitions.

Definition 10 (Support)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a prioritized OrBAC instance. The support of the concrete permission* ls-permitted(s,α,o,$\{w_1, w_2, w_3, w_4\}$) is the minimal subset of rules required to derive it, i.e., the set of rules {Permission(org,r,a,v,c), Employ(org,s,r,w_1), Use(org,o,v,w_2), Consider(org,α,a,w_3), Define(org,s,α,o,c,w_4)}.

Note that a given concrete permission ls-permitted(s,α,o) may have multiple supports, and computing all of its supports can be done by executing a conjunctive query on the instance.

Proposition 3. *Computing all the supports of the concrete permission* ls-permitted(s,α,o) is a tractable task.

Proposition 3 can be achieved using the following query:
$Q_2(\text{r, a, v, c}, w_1, \ldots, w_4)$ = Permission(org,r,a,v,c) $\wedge$ Employ(org,s,r,w_1) $\wedge$ Use(org,o,v,w_2) $\wedge$ Consider(org,α,a,w_3) $\wedge$ Define(org,s,α,o,c,w_4)

The dominance relation extends the partial order defined over the priorities into a partial order defined over subsets of an instance. It was initially introduced in the context of partially ordered DL-Lite ontologies [20].

Definition 11 (Dominance)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a partially ordered OrBAC instance and $\rhd$ be the partial order defined over the priorities in $\mathcal{L}$. Let $\mathcal{S}_1 \subseteq \mathcal{U} \cup \mathcal{P}$ and $\mathcal{S}_2 \subseteq \mathcal{U} \cup \mathcal{P}$. We say that $\mathcal{S}_1$ dominates $\mathcal{S}_2$ if $\forall T_1(\cdot, w_i) \in \mathcal{S}_1$, $\exists T_2(\cdot, w_j) \in \mathcal{S}_2$ such that $w_i \rhd w_j$. Where $T_1(\cdot, w_i), T_2(\cdot, w_j) \in \mathcal{U}$.*

Newt, we introduce the notion of accepted permissions.

Proposition 4 (Accepted permission)**.** *Let $\mathcal{O}_\mathcal{L} = \langle \mathcal{B}, \mathcal{U}, \mathcal{P}, \mathcal{L} \rangle$ be a partially ordered OrBAC instance. The query* ls-permitted(s,α,o)? is granted if $\forall \mathcal{C}$ a conflict of $\mathcal{O}_\mathcal{L}$ (as per Definition 5), $\exists \mathcal{S} \subseteq \mathcal{U} \cup \mathcal{P}$ such that:

1) $\mathcal{S}$ is a support of ls-permitted(s,α,o,$\{w_1, w_2, w_3, w_4\}$) (as per Definition 10), and
2) $\mathcal{S}$ dominates $\mathcal{C}$ (as per Definition 11).

Proposition 5. *Checking whether a given access query is accepted using Proposition 4 is a tractable task.*

C. Example

The example is a scenario of access control in a health care unit. We consider the OrBAC instance $\mathcal{O}_\mathcal{L}$ that is defined over the subject {Mary}, the object {Alex-records}, the action {read}, the roles {anesthetist, nurse, relative}, the view {chronic-records}, the activity {consult} and the contexts {surgery, default}. We assume a partial order that results from ordering roles and contexts separately in the instance, with roles and contexts being incomparable. The priorities w_2, w_1 are assigned to the contexts surgery and default respectively, and the priorities u_3, u_2, u_1 are assigned to the roles anesthetist, nurse and relative respectively. The partial order is given by:

$$\mathbb{1} \rhd w_2 \rhd w_1 \qquad \mathbb{1} \rhd u_3 \rhd u_2 \rhd u_1$$

The different rules of the instance are the following:

$\phi_1 =$ Permission(Hcu, anesthetist, consult, chronic-records, surgery)
$\phi_2 =$ Prohibition(Hcu, nurse, consult, chronic-records, default)
$\phi_3 =$ Prohibition(Hcu, relative, consult, chronic-records, default)
$\varphi_1 =$ Consider(Hcu, read, consult, $\mathbb{1}$)
$\varphi_2 =$ Use(Hcu, Alex-records, chronic-records, $\mathbb{1}$)
$\varphi_3 =$ Employ(Hcu, Mary, anesthetist, u_3)
$\varphi_4 =$ Employ(Hcu, Mary, nurse, u_2)
$\varphi_5 =$ Employ(Hcu, Mary, relative, u_1)
$\varphi_6 =$ Define(Hcu, Mary, read, Alex-records, surgery, w_2)
$\varphi_7 =$ Define(Hcu, Mary, read, Alex-records, default, w_1)

Using Definition 4, the concrete privileges that can be derived from the above OrBAC policy are:

- $\mathcal{O}_\mathcal{L} \models$ ls-permitted(Mary, read, Alex-records, $\{\mathbb{1}, u_3, w_2\}$)
- $\mathcal{O}_\mathcal{L} \models$ ls-prohibited(Mary, read, Alex-records, $\{\mathbb{1}, u_2, w_1\}$)
- $\mathcal{O}_\mathcal{L} \models$ ls-prohibited(Mary, read, Alex-records, $\{\mathbb{1}, u_1, w_1\}$)

a) Conflicts: One can see that the instance is inconsistent since it derives concrete permission and prohibition for the same concrete elements. The conflicts are:

- $\mathcal{C}_1 = \{\phi_1, \phi_2, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_6, \varphi_7\}$
- $\mathcal{C}_2 = \{\phi_1, \phi_3, \varphi_1, \varphi_2, \varphi_3, \varphi_5, \varphi_6, \varphi_7\}$

b) Accepted permission: Consider the following access query: ls-permitted(Mary, read, Alex-records)?. The query has a single support in the policy $\mathcal{S} = \{\phi_1, \varphi_1, \varphi_2, \varphi_3, \varphi_6\}$, the priorities associated with its uncertain elements form the set $\{\mathbb{1}, u_3, w_2\}$. In order for the query to be accepted, as per the Proposition 4, the support $\mathcal{S}$ must dominate both the conflicts $\mathcal{C}_1$ and $\mathcal{C}_2$. The dominance translates to each uncertain element of the support being more preferred to at least one element from the conflict. This holds for both $\mathcal{C}_1$ and $\mathcal{C}_2$. Figure 1 illustrates the dominance of $\mathcal{S}$ over $\mathcal{C}_1$ and $\mathcal{C}_2$. We conclude that the permission ls-permitted(Mary,read,Alex-records,$\{\mathbb{1}, u_3, w_2\}$) is accepted.

(a) $\mathcal{S}$ dominates $\mathcal{C}_1$ (b) $\mathcal{S}$ dominates $\mathcal{C}_2$

Fig. 1: $\longrightarrow$: the strict preference relation ($\rhd$)

VI. Concluding Discussions

Conflicts in access control models arise when both permission and denial rules are used. This paper proposes mechanisms for handling conflicts, without rewriting the security policy or adding any constraints or conditions. Instead, conflicts are addressed when answering access queries. This approach is beneficial for dynamic environments and supports policy evolution as well as efficient query answering.

Priorities may induce a preference relation over the rules to resolve conflicts. Furthermore, the existing methods tend to define more constraints when there is no preference.

Our approach considers a new version of prioritized OrBAC, and distinguishes between fully certain and uncertain rules. It is compatible with the framework of possibility theory for representing uncertainty. It is inspired from inconsistency-tolerant semantics in the context of formal ontologies.

We consider abstract rules as fully certain, and deal with conflicts as a result of connecting abstract elements to concrete elements. We propose two methods for handling partially ordered OrBAC. The query-oriented method consists in defining decision rules to conclude whether a request for a privilege is granted or not. The total order version is trivially efficient. Meanwhile, the partial order version considers total order extension of the partial order, a computationally challenging step. To resolve this issue, the repair-based method adapts the notion of accepted permission from possibilistic lightweight ontology repairs, to avoid enumerating all the total orders that extend a partial order. The accepted permission method considers all the conflicts of an instance in order to make decisions on access queries. This approach is cautious, which is favourable for this kind of application.

We opted for OrBAC because it is a well-formalized, dynamic and contextual model. The proposed methods can be extended to other access control models. For instance, The Role-Based Access Control can be adapted since roles are already considered in OrBAC. However, other notions must be taken into consideration, like hierarchies. The Attribute-Based Access Control is also an interesting model for this kind of applications. However, defining the separation of fully certain and uncertain rules in this model is not straightforward and may require introducing new rules or modifying the model.

In future work, we plan to consider other definitions of conflicts. For instance, two roles may be disjoint and lead to a conflict between two Employ rules for the same subject. We also plan to automate priority assignment in the model.

Acknowledgements: This research was supported by the European Union's Horizon research and innovation program under the MSCA-SE (Marie Skłodowska-Curie Actions Staff Exchange) Call: HORIZON-MSCA-2021-SE-01; Project title: STARWARS [grant agreement 101086252]. Ahmed Laouar's PhD is supported by the French national project ANR Vivah [grant number ANR-20-THIA-0004]. This research has also received support from the ANR project EXPIDA [grant number ANR-22-CE23-0017] and from the ANR project CROQUIS [grant number ANR-21-CE23-0004].
The authors would like to thank the reviewers for their useful comments.

References

[1] Abou El Kalam, A., Deswarte, Y.: Multi-orbac: A new access control model for distributed, heterogeneous and collaborative systems. In: Proceedings of the IEEE Symposium on Systems and Information Security (2006)

[2] Adelin, R., Nugier, C., Alata, É., Nicomette, V., Migliore, V., Kaâniche, M.: Facing emerging challenges in connected vehicles: a formally proven, legislation compliant, and post-quantum ready security protocol. Journal of Computer Virology and Hacking Techniques **18**(4), 425–452 (2022)

[3] Baget, J., Benferhat, S., Bouraoui, Z., Croitoru, M., Mugnier, M., Papini, O., Rocher, S., Tabia, K.: A general modifier-based framework for inconsistency-tolerant query answering. In: Principles of Knowledge Representation and Reasoning (KR), Cape Town, South Africa. pp. 513–516 (2016)

[4] Belabbes, S., Benferhat, S.: Computing a possibility theory repair for partially preordered inconsistent ontologies. IEEE Transactions on Fuzzy Systems pp. 1–10 (2021)

[5] Belabbes, S., Benferhat, S., Chomicki, J.: Elect: An inconsistency handling approach for partially preordered lightweight ontologies. In: Logic Programming and Nonmonotonic Reasoning (LPNMR), Philadelphia, USA. pp. 210–223 (2019)

[6] Benferhat, S., Bouriche, K., Ouzarf, M.: On the possibilistic handling of priorities in access control models. In: Foundations and Applications of Intelligent Systems: Proceedings of the Seventh International Conference on Intelligent Systems and Knowledge Engineering, Beijing, China, Dec 2012 (ISKE 2012). pp. 275–285. Springer (2014)

[7] Benferhat, S., El Baida, R.: A prioritized-based approach to handling conflicts in access control. In: 16th IEEE International Conference on Tools with Artificial Intelligence. pp. 286–293. IEEE (2004)

[8] Benferhat, S., El Baida, R., Cuppens, F.: A stratification-based approach for handling conflicts in access control. In: Proceedings of the eighth ACM symposium on Access control models and technologies. pp. 189–195 (2003)

[9] Bienvenu, M., Bourgaux, C.: Inconsistency-tolerant querying of description logic knowledge bases. Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering: 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures 12 pp. 156–202 (2017)

[10] Boella, G., Hulstijn, J., van der Torre, L.: Argument games for interactive access control. In: The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05). pp. 751–754. IEEE (2005)

[11] Bouij-Pasquier, I., Ouahman, A.A., Abou El Kalam, A., de Montfort, M.O.: Smartorbac security and privacy in the internet of things. In: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA). pp. 1–8. IEEE (2015)

[12] Bringhenti, D., Seno, L., Valenza, F.: An optimized approach for assisted firewall anomaly resolution. IEEE Access **11**, 119693–119710 (2023)

[13] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning **39**(3), 385–429 (2007)

[14] Chouchani, N., Debbech, S., Perin, M.: Model-based safety engineering for autonomous train map. Journal of Systems and Software **183**, 111082 (2022)

[15] Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science **186**, 3–26 (2007). https://doi.org/10.1016/j.entcs.2007.01.064, proceedings of the First Workshop in Information and Computer Security (ICS 2006)

[16] Hasani, S.M., Modiri, N.: Criteria specifications for the comparison and evaluation of access control models. International Journal of Computer Network and Information Security **5**(5), 19 (2013)

[17] Jabal, A.A., Davari, M., Bertino, E., Makaya, C., Calo, S., Verma, D., Russo, A., Williams, C.: Methods and tools for policy analysis. ACM Computing Surveys (CSUR) **51**(6), 1–35 (2019)

[18] Kalam, A.A.E., Baida, R.E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks. pp. 120–131. IEEE (2003)

[19] Laamech, N., Munier, M., Pham, C.: Idsm-o: An iot data sharing management ontology for data governance. In: Proceedings of the 14th International Conference on Management of Digital EcoSystems. pp. 88–95 (2022)

[20] Laouar, A., Belabbes, S., Benferhat, S.: Tractable closure-based possibilistic repair for partially ordered DL-Lite ontologies. In: European Conference on Logics in Artificial Intelligence. pp. 353–368. Springer (2023). https://doi.org/10.1007/978-3-031-43619-2_25

[21] Mohamed, A.K.Y.S., Auer, D., Hofer, D., Küng, J.: A systematic literature review for authorization and access control: definitions, strategies and models. International Journal of Web Information Systems **18**(2/3), 156–180 (2022)