International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 25, No. 3, June, 2020, pp. 285-294
Journal homepage: http://iieta.org/journals/isi

Detection of Conflicts Between Resource Authorization Rules in Extensible Access Control Markup Language Based on Dynamic Description Logic

Shaoyu Yang[1*], Cong Tan[2]

[1] North China University of Water Resources and Electric Power, Zhengzhou 450000, China
[2] Henan University of Animal Husbandry and Economy, Zhengzhou 450000, China

Corresponding Author Email: ysy@ncwu.edu.cn

https://doi.org/10.18280/isi.250301

Received: 17 February 2020
Accepted: 29 April 2020

Keywords:
dynamic description logic (DDL), extensible access control markup language (XACML), access control rule (ACR), rule conflict detection

ABSTRACT

For resources in an open environment, the access control rules (ACRs), which are described by extensible access control markup language (XACML), might have conflicts between each other. To improve the rule management, the root causes of rule conflicts must be identified. This paper firstly formally models the resource attributes by dynamic description logic (DDL), and then investigates inference problems like attribute consistency and rule satisfiability by setting up concept, instance and action knowledge bases. Next, DDL-based rule conflict detection algorithms were designed to identify possible rule conflicts. Finally, the feasibility and decidability of the proposed algorithms were verified through experiments on expanded Continue dataset. The research results provide new insights to the detection of conflicts between resource authorization rules (RARs).

1. INTRODUCTION

In an open environment, resources are often scheduled and accessed through the collaboration and combination among multiple organizations and systems. The access control rules (ACRs) for descriptive attributes of inter-domain resources need to be maintained and updated by decision makers from other domains. This cross-domain resource access control mode raises new requirements for ACR formulation and verification.

For one thing, the knowledge sharing and permission coverage might exist in the conceptual structure and correlations of resource attributes. For another, the intra-domain resources could be combined and migrated at any time, and ACR authorization and revocation add difficulty to rule management, pushing up the possibility of conflicts between resource authorization rules (RARs).

Most access control systems provide mitigating methods for the above-mentioned rule conflicts, namely, prioritizing "allow", "deny", or "use" permissions. Focusing only on the final judgement of the rules, these mitigation methods merely ensure the determinism of the evaluation for access control system. For resource visitors and rule makers, the process and causes of rule conflicts are not available, making it impossible to find the source of conflicts from the structure of resource attributes and RARs.

Considering the RAR conflicts in a distributed environment, this paper probes into the mitigating methods adopted in extensible access control markup language (XACML). The ACRs were formally expressed by dynamic description logic (DDL). The possible RAR conflicts were verified through model inference.

2. LITERATURE REVIEW

As a description standard for RARs, the XACML [1] was formulated by Organization of the Advancement of Structured Information Standard (OASIS) to promote the consistency of ACR descriptions on the network layer. With a fine-grained description method for attribute authorization and an easily expandable form of rule description, the XACML offers a suitable tool to describe the access control and authorization of distributed resources.

In recent years, the XACML has been frequently adopted for logic description and reasoning of ACRs in distributed environments. Facing the heterogeneous Internet of things (IoT), Atlam et al. [2] depicted the ACRs of lightweight IoT devices with the XACML, and designed a distributed and flexible cross-domain resource access control method. Based on the grammar rules of the XACML, the designed method expands the application scope and enhances the fault tolerance and authorization validation of the original model. Focusing on cloud service application scenarios in health services, Ayache et al. [3] developed an XACML-based cloud service for verifying ACRs, which realizes data sharing, task invocation and activity coordination across service domains. Kanwal et al. [4] tackled the security of data release and sharing of electronic medical records in a hybrid cloud environment, made fine-grained XACML description of resource ACRs in that environment, and thereby improved the capabilities of the ACRs in privacy protection and access control. For safe access to IoT devices in distributed network, Charaf et al. [5] proposed an XACML-based access control method for terminal devices, which is reliable, available and confidential.

Damiano et al. [6] applied blockchain technology to describe ACRs and establish a decentralized trust verification mechanism for cross-domain resources. Specifically, the security performance of resource attributes was subject to fine-grained description, and the attributes and rules were linked to the blockchain in the form of smart contract. The proposed mechanism was validated in the Ethereum environment. Based on the blockchain technology, Ma et al. [7] created a distributed key management architecture for the hierarchical access control of the IoT. The architecture satisfies the demand for access control of cross-domain cloud services, supports decentralized and fine-grained verification of attributes, and integrates privacy protection into the ACRs. Zyskind and Nathan [8] combined blockchain technology and off-chain storage into a data management platform for privacy protection. In the platform, the access control process mainly focuses data reading, failing to form a complete access control system. Sukhodolskiy and Zapechnikov [9] put forward an access control method for data storage in a cloud environment. Wang et al. [10] realized data storage and sharing without the participation of providers, using Ethereum and an attribute-based access control method. Furthermore, many other researchers have studied the access control strategy described by XACML about Policies formalization [11], Automatic Testing [12], Model testing [13], policy tracing [14], automated fault localization [15].

The above studies mainly explored the access control mechanism of cross-domain resources in different application environments, and developed some mitigation methods for rule conflicts based on the XACML. Most of these methods mitigate the impact of rule conflicts on the authorization process from the perspective of authorization results. However, the nature of the mitigation has not been analyzed from the angles of rule description and attribute structure, weakening the ability of reasoning and verification. The XACML-based rule combination algorithms neglect the attributes and the underlying causes leading to rule conflicts, and overlook the influence of rule correlations and attribute hierarchy on the rule conflicts. To make up for the gaps, the rule conflicts should be detected and mitigated before judging the authorization, using the reasoning ability of dynamic description logic (DDL). In this way, the security management personnel in the domain can discover rule conflicts in time, find the reasons of conflicts, and simplify the relevant rules.

3. DDL-BASED AUTHORIZATION AND REASONING MODEL

The attributes of cloud resources have an abundance of semantic ontology descriptions, making the authorization reasoning between resources a possibility. Here, the ontology language of resource attributes is mapped to the DDL language environment, and the resource authorization is formally verified by the DDL reasoning mechanism.

3.1 Attribute-based access control model

The modelling of attribute access control mainly describes the knowledge of TBox, ABox, and ActBox. As shown in Figure 1, the attribute-based fine-grained access control model encompasses a concept sub-model, an instance sub-model, and an action sub-model.

The TBox knowledge uses axioms to systematically describe the conceptual structure of ontology resources. The structured knowledge can depict the inheritance and inclusion semantics expressed by attributes.

The ABox knowledge mainly verifies whether the conceptual implication relationship (CIR) of instance attributes is consistent.

The ActBox knowledge mainly describes the necessary conditions, formula set, and result set for resource authorization. The actions in the ActBox fall into atomic authorization action, combined authorization action and transfer authorization action.

3.1.1 Concept sub-model

The hierarchical conceptual knowledge in TBox lays the basis for judging the completeness of resource instance set. This subsection defines the concept sub-model through formal description, paving the way for DDL reasoning.

Definition 1. Concept sub-model $\mathcal{M}_{Concept}$: For TBox T in knowledge base KB, there exists a concept sub-model $\mathcal{M}_{Concept}(A_1, A_2, \dots) \vDash$ T.

(1) If and only if any concept in T contains axiom $A \equiv A'$, there exists an interpretation *I(ω)* that satisfies $A^{I(\omega)} \equiv A'^{I(\omega)}$ in any space ω of the world W.

(2) If and only if any relationship in T contains axiom $R \sqsubseteq R'$, there exists an interpretation *I(ω)* that satisfies $R^{I(\omega)} \subseteq R'^{I(\omega)}$ in any space ω of the world W.

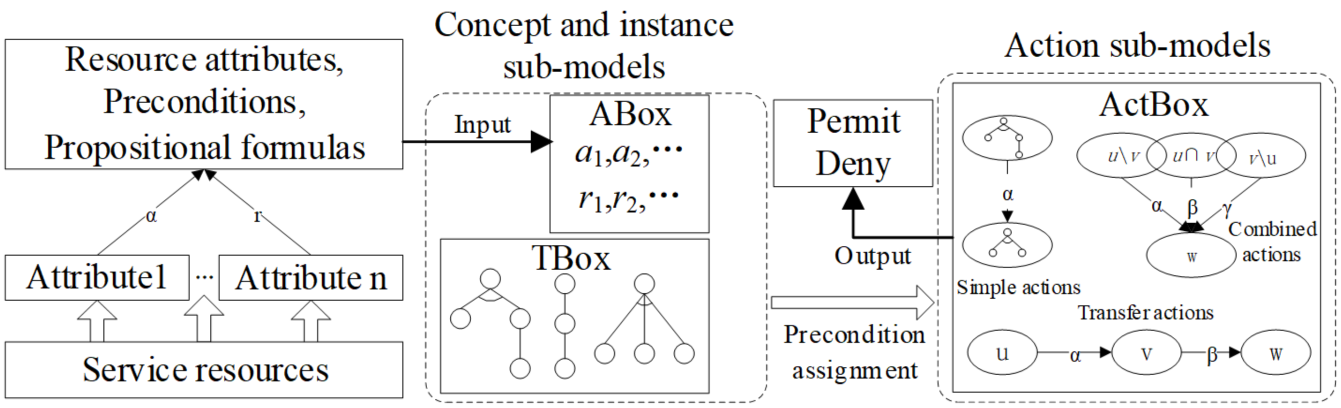

Figure 1. The attributed-based fine-grained access control model

3.1.2 Instance sub-model

The resource instances in ABox are formal resource descriptions based on the ontology resource model, providing the basic information of resources. These instances are essential to judging the realizability of authorization actions. Hence, it is necessary to verify the satisfiability of the concept sub-model for each instance. The instance sub-model is defined as follows:

Definition 2. Instance sub-model $\mathcal{M}_{Instance}$: For ABox A and TBox T in knowledge base KB, there exists an instance sub-model $\mathcal{M}_{Instance}(a_1, a_2, \dots) \vDash \mathrm{T}$. If and only if $a_1 \in A$, *I(A)* is an instance sub-model $\mathcal{M}_{Instance}$ in *T*.

3.1.3 Action sub-model

The action sub-model is the abstract description of fine-grained access control and authorization. Each authorization action is formally depicted as a transition between instance sub-models realized by the assignment of a formula set. With the action sub-model, the decision-maker can assign values from the instance set in ABox to each action, judge the realizability of the action, and determine whether the resources are suitable for authorization access.

Definition 3. Action sub-model $\mathcal{M}_{Action}$: According to the authorization rules in ActBox ACT, there exists an assignment $\Upsilon^{\mathcal{F}}$ in the precondition set $\mathcal{F}$ that makes instance sub-model $\mathcal{M}_{Instance_P}$ satisfy the preconditions of action $\alpha = (\mathcal{F}, \mathcal{M}_{Instance_P}/\mathcal{M}_{Instance_E})$, and transfers the result into another instance sub-model $\mathcal{M}_{Instance_E}$. The action sub-model can be denoted as $\mathcal{M}_{Action}(\alpha, \beta \dots) \vDash \exists \Upsilon^{\mathcal{F}}, \mathcal{M}_{Instance_P} \rightarrow_{\alpha}^{\Upsilon} \mathcal{M}_{Instance_E}$.

3.2 Reasoning based on the authorization model

The preceding subsection models the authorization in resource access control, using TBox, ABox, and ActBox in knowledge base KB. The established authorization model can be reasoned based on the inference engine.

During the authorization, the reasoning task mainly checks the consistency of concepts and instances, the realizability of authorization actions, and the containment between actions. The consistency check ensures that the instances in each ABox satisfy the attributes and attribute correlations of conceptual knowledge structure. The realizability check reflects whether an authorization action on resources is achievable. The containment check clarifies the partial ordering of authorization actions among resources, and helps to reduce redundant RARs. The consistency, realizability, and satisfiability in model reasoning are defined as follows:

3.2.1 Consistency of concepts and instances

The consistency reasoning targets the instances and concepts in ABox and TBox under the static scene. The reasoning deals with the satisfiability of concepts in TBox, and the instance checking and consistency judgement of ABox instances under the constraint of TBox.

Definition 4. Concept satisfiability: Any concept A in concept sub-model $\mathcal{M}_{Concept}$ is satisfiable, if there exists an interpretation $I^{\mathcal{M}_{Concept}} \neq \emptyset$ for that concept in the model.

Definition 5. Instance checking: For instance sub-model $\mathcal{M}_{Instance}$, there exists an interpretation $I^{\mathcal{M}_{Instance}} \neq \emptyset$ of instance $C(a)$ about the model in ABox A.

Definition 6. Instance consistency: For instance sub-model $\mathcal{M}_{Instance}$, there exist two instances $C(a), C \in TBox\ T$. The instance consistency can be denoted as $A^{I(T)} \vDash C(a)$.

For a given instance ABox A, it should first satisfy the requirement of instance checking, that is, there exists a concept that matches this instance. Next, the concept should be satisfiable, as evidenced by the interpretation of TBox in the concept sub-model. Then, the instance consistency can be checked by detecting the conflicts between ABox and the concepts in TBox.

3.2.2 Realizability

The realizability check of authorization actions is a dynamic reasoning in access control. The realizability of complex actions can be extended from the realizability of atomic authorization actions.

Definition 7. Realizability of authorization actions: The atomic authorization action α is realizable, if the instance sub-model $\mathcal{M}_{Instance} \subseteq I(u) = (\Delta, \cdot^{I(u)})$ and concept sub-model $\mathcal{M}_{Concept} \subseteq I(v) = (\Delta, \cdot^{I(v)})$ in the initial ABox A description and TBox T satisfy $u \rightarrow_{\mathcal{F}}^{\mathcal{M}_{Instance}} v$. The realizability can be denoted as $Rel_{I(u)}^{T}(\alpha)$.

The realizability of an atomic authorization action contains the satisfiability of the preconditions and the consistency of the instances in ABox. The ABox instances in interpretation $I(u)$ can be judged by the instance consistency check, and those in interpretation $I(v)$ can be judged by the concept consistency check in TBox. Then, it can be determined intuitively that there is no inconsistency between the preconditions and the consequences of the authorization action.

The realizability of complex actions can be described as follows:

For combined authorization actions, the realizability can be expressed as: $Rel_{I(W)}^{T}(PA) \Rightarrow Rel_{I(u)}^{T}(\alpha) \vee Rel_{I(u')}^{T}(\beta) \vee \cdots$;

For transfer authorization actions, the realizability can be expressed as: $Rel_{I(W)}^{T}(PA) \Rightarrow Rel_{I(u)}^{T}(\alpha) \wedge Rel_{I(u')}^{T}(\beta) \wedge \cdots$.

3.2.3 Satisfiability

The satisfiability of an authorization action is mainly judged based on the satisfiability of the precondition set of the action. The latter is verified against the given TBox, ABox, and ActBox.

Definition 8. Satisfiability of authorization actions: For the given TBox, ABox, and ActBox, an action α with a precondition or precondition set $\mathcal{F}$ is satisfiable if:

The atomic action with $\mathcal{F}$ as the precondition is realizable in T and A;

There exists a possible space ω making $(\mathcal{M}, \omega) \vDash \mathcal{F}$.

4. XACML-BASED RULE CONFLICT DETECTION

In the XACML-based resource authorization framework, there are two kinds of assignments, namely, permit and deny. Permit allows the subject to acquire the resource access right, and deny rejects the subject's access request. Each request for resource access may have different authorization results, causing rule conflicts [16-18].

From the hierarchical inheritance of attribute concepts, this section analyzes the causes of rule conflicts. The attribute-based fine-grained access control mode was adopted to derive the CIRs in attribute hierarchy, and thus detect rule conflicts.

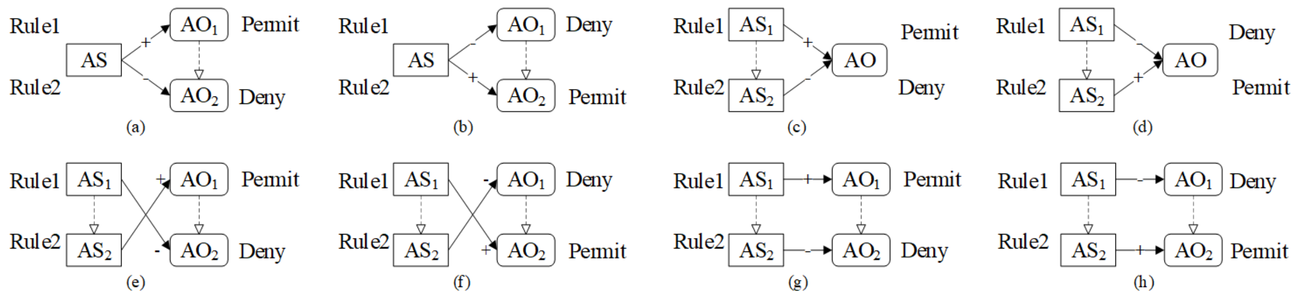

Figure 2. The hierarchy of RAR conflicts

4.1 Types of rule conflicts

The rule authorization effect depends on whether the subject is permitted to access or denied from accessing the requested resource. For the attributes of the subject, rule conflicts may arise from the CIRs in concepts and instances; for the attributes of the resource, rule conflicts may arise from the CIRs. The potential rule conflicts are classified as Figure 2, where AS is subject attribute, AO is resource attribute, hollow arrow is the implication and inheritance between attribute concepts, the plus sign is permit, and the minus sign is denied.

For subject attributes, the lower attributes inherit the rights of the upper attributes. For resource attributes, the lower attributes are the fine-grained expression of the upper attributes. As shown in Figure 2, the RARs were divided into eight types (a)-(h) according to the CIRs of subjects and resources, the granularity of resource attributes, and the requirements on permit and deny. The difference between RAR types comes from the hierarchy of conceptual knowledge. The following analyzes the possible rule conflicts in each type of RARs. Note that *Rule*1 and *Rule*2 are denoted as action sub-models $\mathcal{M}_{Act_1}$ and $\mathcal{M}_{Act_2}$, respectively.

$$\text{(a) } \mathcal{M}_{Act_1} \vDash AS \rightarrow_{\alpha+}^{\gamma} AO_1, \mathcal{M}_{Act_2} \vDash AS \rightarrow_{\alpha-}^{\gamma} AO_2$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \nRightarrow \mathcal{M}'_{Act_1} \vDash AS \rightarrow_{\alpha+}^{\gamma} AO_2 \\ \mathcal{M}_{Act_2} \vDash AS \rightarrow_{\alpha-}^{\gamma} AO_2 \end{matrix}\right\} \nRightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

In Figure 2, (a) and (b) describe the authorization of hierarchical resources to the same subject. *AS* means the permit for coarse-grained attribute AO_1. The permit cannot be inherited by the fine-grained attribute AO_2, that is, there exists no interpretation of action sub-model $\mathcal{M}'_{Act_1}$ that meets the permit for AO_2. Therefore, the two RAR actions will not have any conflict.

$$\text{(b) } \mathcal{M}_{Act_1} \vDash AS \rightarrow_{\alpha-}^{\gamma} AO_1, \mathcal{M}_{Act_2} \vDash AS \rightarrow_{\alpha+}^{\gamma} AO_2$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS \rightarrow_{\alpha-}^{\gamma} AO_2 \\ \mathcal{M}_{Act_2} \vDash AS \rightarrow_{\alpha+}^{\gamma} AO_2 \end{matrix}\right\} \Rightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

If the access to coarse-grained resource attributes is denied, the deny will be inherited by the lower fine-grained resource attributes, that is, there exists an interpretation of action sub-model $\mathcal{M}'_{Act_1}$ to meet the deny of fine-grained attribute AO_2. In this case, the RARs will conflict each other.

$$\text{(c) } \mathcal{M}_{Act_1} \vDash AS_1 \rightarrow_{\alpha+}^{\gamma} AO, \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO \\ \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO \end{matrix}\right\} \Rightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

(c) and (d) are the resource authorization of subject attributes with CIRs. In the case of (c), the permit of AS_1 for AO can be transferred to its sub-attribute AS_2, such that there exists an interpretation of action sub-model $\mathcal{M}'_{Act_1}$ that permits AS_2 to access AO. This conflicts with the deny of the other rule $\mathcal{M}_{Act_2}$.

$$\text{(d) } \mathcal{M}_{Act_1} \vDash AS_1 \rightarrow_{\alpha-}^{\gamma} AO, \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO \\ \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO \end{matrix}\right\} \Rightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

Both (c) and (d) are resulted from the authorization inheritance of subject and resource attributes. In the case of (d), the deny of AS_1 for AO leads to the deny of AS_2 for AO, that is, there exists an interpretation of action sub-model $\mathcal{M}'_{Act_1}$ that denies AS_2 from accessing AO. This conflicts with the permit of the other rule $\mathcal{M}_{Act_2}$.

$$\text{(e) } \mathcal{M}_{Act_1} \vDash AS_1 \rightarrow_{\alpha-}^{\gamma} AO_2, \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO_1$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO_2 \\ \mathcal{M}_{Act_2} \nRightarrow \mathcal{M}'_{Act_2} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO_2 \end{matrix}\right\} \nRightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

(a)-(d) are four basic RARs of atomic level attributes. Considering more complex situations, (e) describes the cross-authorization of hierarchical subjects and resources. The subject attributes can inherit the authorization effect of the upper attributes according to the CIRs. Hence, there exists an $\mathcal{M}'_{Act_1}$ that denies AS_2 from accessing AO_2. Since the permit for a coarse-grained resource attribute cannot be inherited by lower fine-grained attribute, there is no $\mathcal{M}'_{Act_2}$ that describes the permit for AO_2. In this case, the two rules will not have any conflict.

$$\text{(f) } \mathcal{M}_{Act_1} \vDash AS_1 \rightarrow_{\alpha+}^{\gamma} AO_2, \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO_1$$
$$\left\{\begin{matrix} \mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \rightarrow_{\alpha+}^{\gamma} AO_2 \\ \mathcal{M}_{Act_2} \Rightarrow \mathcal{M}'_{Act_2} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO_2 \end{matrix}\right\} \Rightarrow Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$$

The cross-authorization of (f) is opposite to the effect of (e). In this case, the CIR of each subject attribute make it possible to transfer permit to sub-attribute AS_2. Hence, there exists an $\mathcal{M}'_{Act_1}$ that allows AS_2 to access AO_2. Meanwhile, the deny for coarse-grained resource attribute AO_1 can be inherited by lower fine-grained attribute AO_2. Thus, there exists an $\mathcal{M}'_{Act_2}$ that denies AS_2 from accessing AO_2. As a result, the two rules will have conflict.

$$\text{(g) } \mathcal{M}_{Act_1} \vDash AS_1 \rightarrow_{\alpha+}^{\gamma} AO_1, \mathcal{M}_{Act_2} \vDash AS_2 \rightarrow_{\alpha-}^{\gamma} AO_2$$

$$\left.\begin{cases}\mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \to^{\gamma}_{\alpha^+} AO_1 \\ \mathcal{M}_{Act_2} \vDash AS_2 \to^{\gamma}_{\alpha^-} AO_2\end{cases}\right\} \nRightarrow Conflict^{\mathcal{M}_{Act_2}}_{\mathcal{M}_{Act_1}}$$

(g) and (h) are two parallel authorizations. (g) describes the different authorization effects between upper resource attributes to upper subject attributes. The subject attribute AS_1 transfers the permit for AO_1 to the lower sub-attribute AS_2 via the CIR. Hence, there exists an $\mathcal{M}'_{Act_1}$ that allows AS_2 to access AO_1. For action sub-models $\mathcal{M}'_{Act_1}$ and $\mathcal{M}_{Act_2}$, (g) can be transformed into atomic authorization (a). Therefore, the two rules will have no conflict.

(h) $\mathcal{M}_{Act_1} \vDash AS_1 \to^{\gamma}_{\alpha^-} AO_1, \mathcal{M}_{Act_2} \vDash AS_2 \to^{\gamma}_{\alpha^+} AO_2$

$$\left.\begin{cases}\mathcal{M}_{Act_1} \Rightarrow \mathcal{M}'_{Act_1} \vDash AS_2 \to^{\gamma}_{\alpha^-} AO_1 \\ \mathcal{M}_{Act_2} \vDash AS_2 \to^{\gamma}_{\alpha^+} AO_2\end{cases}\right\} \Rightarrow Conflict^{\mathcal{M}_{Act_2}}_{\mathcal{M}_{Act_1}}$$

Similar to the conflict analysis of (g), the subject attribute AS_1 in the case of (h) transfers the deny for AO_1 to the lower sub-attribute AS_2 via the CIR. Hence, there exists an $\mathcal{M}'_{Act_1}$ that denies AS_2 from accessing AO_1. For action sub-models $\mathcal{M}'_{Act_1}$ and $\mathcal{M}_{Act_2}$, (h) can be transformed into (b). Therefore, the two rules will have conflict.

4.2 DDL-based conflict detection

From the above analysis on the types of rule conflicts, the possible situations of rule conflicts can be divided into four kinds of atomic authorizations (a)-(d). Based on the subject CIRs and inheritance of resource granularity, the cases (e)-(h) could be transformed into (a)-(b).

Among the four cases, no conflict will occur in (a), for the permit for coarse-grained resource attribute AO_1 cannot be transferred to lower fine-grained attribute AO_2. In the other three cases, rule conflicts may occur due to the presence of CIR and deny transfer to fine-grained attributes. To detect the conflicts between XACML rules, two issues must be taken into account: the CIRs and type of authorization between subject AS and resource AO.

To facilitate the description of the permit and deny in XACML rules, the action sub-model was divided into a set of positive interpretations $\mathcal{M}^{Permit}_{Act} \equiv AS \to^{\gamma}_{\alpha^+} AO$ and a set of negative interpretations $\mathcal{M}^{Deny}_{Act} \equiv AS \to^{\gamma}_{\alpha^-} AO$. Depending on the authorization effect, the RAR with resource is included in different interpretation sets. In case (a), $\mathcal{M}_{Act_1} \in \mathcal{M}^{Permit}_{Act}$ and $\mathcal{M}_{Act_2} \in \mathcal{M}^{Deny}_{Act}$. Besides, the subject attributes needed for each authorization action were represented by an interpretation of an instance sub-model $\mathcal{M}^{\mathcal{F}}_{instance_P}$, where $\mathcal{F}$ is the precondition made up of subject attributes. The authorized resource attributes were represented by another instance sub-model $\mathcal{M}_{instance_E}$. Next, the rule conflicts were detected from subject and resource attributes, respectively.

The inputs of Algorithm 1 include the positive RAR set $\mathcal{M}^{Permit}_{Act}$ and the negative RAR set $\mathcal{M}^{Deny}_{Act}$, as well as the conceptual knowledge base TBox T. The algorithm detects the rule conflicts arising from the hierarchical inheritance of the resource attributes accessed by the subject, and returns a conflict symbol $Conflict^{\mathcal{M}_{Act_2}}_{\mathcal{M}_{Act_1}}$.

In Lines 1 and 2, positive RARs and negative RARs in the rules are traversed. The action sub-models are expressed as $\mathcal{M}_{Act_1}$ and $\mathcal{M}_{Act_2}$. If there exists an assignment $\gamma^{\mathcal{F}}$ that makes instance $\mathcal{M}_{Instance_P}$ satisfy the preconditions of $\mathcal{M}_{Act_1}$ and $\mathcal{M}_{Act_2}$ (Line 3), and if there exist concept sub-models $\mathcal{M}^{I}_{Concept_{E_1}}$ and $\mathcal{M}^{I}_{Concept_{E_2}}$ that satisfy instances among the authorization objects $\mathcal{M}_{Instance_{E_1}}$ and $\mathcal{M}_{Instance_{E_2}}$ of the rules (Line 4), then the satisfiability of the concept sub-models will be judged by TBox T. If TBox T contains the deny for fine-grained resource attributes, i.e. there exists a CIR $\mathcal{M}_{Concept_{E_2}} \sqsubseteq \mathcal{M}_{Concept_{E_1}}$, then the rules will conflict as in case (b).

Algorithm 1. Rule conflict detection based on the hierarchy of resource attributes

```
Inputs: $\mathcal{M}^{Permit}_{Act}$;
  $\mathcal{M}^{Deny}_{Act}$;
  TBox T;
Outputs: $Conflict^{\mathcal{M}_{Act_2}}_{\mathcal{M}_{Act_1}}$
1: For each $\mathcal{M}_{Act_1}$ in $\mathcal{M}^{Permit}_{Act}$ do
2:   For each $\mathcal{M}_{Act_2}$ in $\mathcal{M}^{Deny}_{Act}$ do
3: $\exists\gamma^{\mathcal{F}}, \mathcal{M}_{Act_1} \vDash \mathcal{M}_{Instance_P} \to^{\gamma}_{\alpha^-} \mathcal{M}_{Instance_{E_1}}$ AND
       $\mathcal{M}_{Act_2} \vDash \mathcal{M}_{Instance_P} \to^{\gamma}_{\alpha^+} \mathcal{M}_{Instance_{E_2}}$;
4: $\mathcal{M}_{Instance_{E_1}} \vDash \mathcal{M}^{I}_{Concept_{E_1}}, \mathcal{M}_{Instance_{E_2}} \vDash \mathcal{M}^{I}_{Concept_{E_2}}$;
5:   If $\mathcal{M}_{Concept_{E_2}} \sqsubseteq \mathcal{M}_{Concept_{E_1}}$ ,in T then
6:     $ConfictDissolve(\mathcal{M}_{Act_1}, \mathcal{M}_{Act_2})$;
7:     Return true;
8:     Else
9:     Return false;
10:    End if
11:    End for each
12:    End for each
```

Then, the conflicting action sub-models are processed: the permit $\mathcal{M}_{Act_2}$ for fine-grained resource attributes is deleted to terminate the inheritance between resource attributes (Line 6). If there is a permit for coarse-grained resource attributes, or if the rule sets contain no assignment that meets the requirements, then the rule sets do not contain rule conflicts (Case a).

Algorithm 1 mainly detects the RAR conflicts arising from the inheritance between resource attributes (Cases a and b). The algorithm locates the conflicting rules, and calls the conflict mitigation function to handle these rules.

TBox provides a hierarchical relationship reflecting the concepts of resource attributes, and determines the existence of rule conflicts by verifying the $\sqsubseteq$ relationship between resource attributes. This CIR belongs to the problem of concept consistency verification in description logic.

The rule conflicts arising from the hierarchy of subject attributes can also be detected by verifying the $\sqsubseteq$ relationship (Algorithm 2).

Similar to those of Algorithm 1, the inputs of Algorithm 2 contain rule sets and the conceptual knowledge base. The algorithm detects the rule conflicts arising from the hierarchy of subject attributes, and returns a conflict symbol.

The algorithm firstly traverses the rule sets (Lines 1 and 2). Under the following conditions, the rules will have conflicts of types (c) and (d): there exist the assignments $\gamma^{\mathcal{F}_1}$ and $\gamma^{\mathcal{F}_2}$ among subject attributes that make instance sub-models $\mathcal{M}_{Instance_{P_1}}$ and $\mathcal{M}_{Instance_{P_2}}$ satisfy the preconditions of $\mathcal{M}_{Act_1}$ and $\mathcal{M}_{Act_2}$, respectively (Line 3); TBox T contains interpretations $\mathcal{M}^{I}_{Concept_{P_1}}$ and $\mathcal{M}^{I}_{Concept_{P_2}}$ that satisfy concept sub-models (Line 4); the concept sub-models have the CIR (Line 5).

Then, the conflicting rules are processed: the lower subject attributes are deleted to keep the consistency between the authorization of higher subject attributes (Line 6). If there is no CIR between subject attributes, then the rule sets will not face any rule conflict arising from the hierarchy of subject attributes.

The above-mentioned atomic RAR conflicts can be detected and resolved by the two algorithms. The complex and cross-authorizations can be handled similarly.

Algorithm 2. Rule conflict detection based on the hierarchy of subject attributes

```
Inputs: $\mathcal{M}_{Act}^{Permit}$;
    $\mathcal{M}_{Act}^{Deny}$;
    $TBox$ T;
Output: $Conflict_{\mathcal{M}_{Act_1}}^{\mathcal{M}_{Act_2}}$
1: For each $\mathcal{M}_{Act_1}$ in $\mathcal{M}_{Act}^{Permit}$ do
2:   For each $\mathcal{M}_{Act_2}$ in $\mathcal{M}_{Act}^{Deny}$ do
3:   $\exists {\gamma_1}^{\mathcal{F}_1}, {\gamma_2}^{\mathcal{F}_2}, \mathcal{M}_{Act_1} \vDash \mathcal{M}_{Instance_{P_1}} \rightarrow_{\alpha^+}^{\gamma_1} \mathcal{M}_{Instance_E}$
AND $\mathcal{M}_{Act_2} \vDash \mathcal{M}_{Instance_{P_2}} \rightarrow_{\alpha^-}^{\gamma_2} \mathcal{M}_{Instance_E}$;
4: $\mathcal{M}_{Instance_{P_1}} \vDash \mathcal{M}_{Concept_{P_1}}^{I}, \mathcal{M}_{Instance_{P_2}} \vDash \mathcal{M}_{Concept_{P_2}}^{I}$;
5:   If $\mathcal{M}_{Concept_{P_2}} \sqsubseteq \mathcal{M}_{Concept_{P_1}}$ in T then
6:     $ConfictDissolve(\mathcal{M}_{Act_1}, \mathcal{M}_{Act_2})$;
7:     Return true;
8:     Else
9:     Return false;
10:    End if
11:  End for each
12:  End for each
```

4.3 DDL-based reasoning

The DDL can reason about the ACR authorization based on the process and transfer, and provide the reasoning ability for the global rule library via the Tableau algorithm. In rule conflict detection, the reasoning mainly covers two aspects: if the rule conflict arises from the hierarchy of attributes, the key is to validate the CIR that interprets the instance sub-model; if the rule conflict arises from RAR transfer, the key is to verify the consistency between the assertion sets of RAR action and instance sub-models. These reasoning problems are comparable to the satisfiability problem and the consistency detection problem in the DDL [19, 20].

Definition 9. Concept sub-models $\mathcal{M}_{Concept_1}$ and $\mathcal{M}_{Concept_2}$ satisfy the CIR $\mathcal{M}_{Concept_1} \sqsubseteq \mathcal{M}_{Concept_2}$, if and only if TBox T contains an interpretation I that makes $\mathcal{M}_{Concept_1}^{I} \subseteq \mathcal{M}_{Concept_2}^{I}$.

Theorem 1. The CIR $\mathcal{M}_{Concept_1} \sqsubseteq_T \mathcal{M}_{Concept_2}$ holds between the concept sub-models in TBox T, if and only if $\mathcal{M}_{Concept_1}^{I} \sqcap \neg\mathcal{M}_{Concept_2}^{I}$ is unsatisfiable, i.e. $\mathcal{M}_{Concept_1}^{I} \sqcap \neg\mathcal{M}_{Concept_2}^{I} = \emptyset$.

If TBox T contains an interpretation I that makes two sub-models $\mathcal{M}_{Concept_1}$ and $\mathcal{M}_{Concept_2}$ satisfy the CIR, and if there exists a concept model $\mathcal{M}_{Concept_1}^{I} \sqcap \neg\mathcal{M}_{Concept_2}^{I} \neq \emptyset$ that satisfies T, then there exists an instance sub-model $\mathcal{M}_{Instance_1}^{I} \in \mathcal{M}_{Concept_1}^{I}$ that satisfies interpretation I. Due to the existence of the CIR, there also exists $\mathcal{M}_{Instance_2}^{I} \in \mathcal{M}_{Concept_2}^{I}$, which contradicts $\mathcal{M}_{Instance_1}^{I} \in \neg\mathcal{M}_{Concept_2}^{I}$. Hence, no such instance exists in $\mathcal{M}_{Instance_1}^{I} \sqcap \neg\mathcal{M}_{Concept_2}^{I}$.

Definition 10. If the two assertions $\mathcal{F}_1$ and $\mathcal{F}_2$ in instance sub-models $\mathcal{M}_{Instance_1}$ and $\mathcal{M}_{Instance_2}$ that transfer RAR results obey $\mathcal{F}_1 \sqcap \mathcal{F}_2 = \emptyset$, then the two instance sub-models are inconsistent, i.e. $\mathcal{M}_{Instance_1} \sqcap \mathcal{M}_{Instance_2} = \emptyset$.

The satisfiability verification of the DDL can be achieved by the Tableaux algorithm [19]. The reasoning verification of two concept or instance sub-models can be realized by the following rules.

Take the satisfiability verification of instance sub-models for example. By the following rules, an expansion set $\mathcal{ES}$ of attributes can be extended from $\mathcal{M}_{Instance_1}$ and $\mathcal{M}_{Instance_2}$. Then, the satisfiability of the two sub-models can be evaluated by judging whether the expansion set brings rule conflicts. If the expansion set contains $\perp$, then the two sub-models are not satisfiable, and face rule conflicts.

$\sqcap$ rule: If there exists an interpretation I in TBox T that makes $\mathcal{M}_{Instance_1}^{I} \sqcap \mathcal{M}_{Instance_2}^{I} \in \mathcal{ES}$ ($\mathcal{M}_{Instance_1}^{I} \notin \mathcal{ES}, \mathcal{M}_{Instance_2}^{I} \notin \mathcal{ES}$), then $\{\mathcal{M}_{Instance_1}, \mathcal{M}_{Instance_2}\}$ should be expanded to $\mathcal{ES}$;

$\sqcup$rule: If there exists an interpretation I in TBox T that makes $\mathcal{M}_{Instance_1}^{I} \sqcup \mathcal{M}_{Instance_2}^{I} \in \mathcal{ES}$ ($\mathcal{M}_{Instance_1}^{I} \notin \mathcal{ES}, \mathcal{M}_{Instance_2}^{I} \notin \mathcal{ES}$), then $\mathcal{M}_{Instance_*}$ should be expanded to $\mathcal{ES}$, where $\mathcal{M}_{Instance_*} = \mathcal{M}_{Instance_1}^{I}$ or $\mathcal{M}_{Instance_*} = \mathcal{M}_{Instance_2}^{I}$;

$\exists$ rule: If there exists an interpretation I in TBox T that makes $\exists R.\mathcal{M}_{Instance_1}^{I} \in \mathcal{ES}$, and $\nexists \mathcal{M}_{Instance_2}^{I}, R(\mathcal{M}_{Instance_1}^{I}, \mathcal{M}_{Instance_2}^{I} \in \mathcal{ES}, \mathcal{M}_{Instance_2}^{I} \in \mathcal{ES})$, then $\{\mathcal{M}_{Instance_2}, R(\mathcal{M}_{Instance_1}, \mathcal{M}_{Instance_2})\}$ should be expanded to $\mathcal{ES}$;

$\forall$ rule: If there exists an interpretation I in TBox T that makes $\forall R.\mathcal{M}_{Instance_1}^{I} \in \mathcal{ES}$, $\mathcal{M}_{Instance_2}^{I} \notin \mathcal{ES}$, then $\{\mathcal{M}_{Instance_2}\}$ should be expanded to $\mathcal{ES}$;

Action α rule: If there exist an interpretation I in TBox T and an assignment set $\gamma^{\mathcal{F}}$ that make $\mathcal{M}_{Instance_1}^{I} \longrightarrow_{\alpha}^{\mathcal{F}} \mathcal{M}_{Instance_2}^{I}$ ($\mathcal{M}_{Instance_2} \in \mathcal{ES}$), then $\mathcal{M}_{Instance_1}$ should be replaced with $\mathcal{M}_{Instance_2}$ in $\mathcal{ES}$.

Theorem 2. Reliability: The DDL-based rule conflict detection is reliable.

The DDL-based rule conflict detection method targets two kinds of rule conflicts in authorization: the conflicts arising from the hierarchy of attributes and those arising from the transfer of authorization.

For the first kind of rule conflicts, the implication and inheritance relationships that stem from the hierarchy of attributes can be converted by Algorithms 1 and 2 into the CIRs in TBox for satisfiability verification. The satisfiability problems can be solved by the Tableaux algorithm in description logic.

For the second kind of rule conflicts, the rule conflict detection can be converted by RAR transfer rules into the satisfiability problem of instance sub-models of the authorization results.

Action α rule is derived from the DDL description and reasoning mechanism. The correctness of the algorithm can be ensured by replacing elements in the extended set. Therefore, the DDL-based rule conflict detection must be correct.

Theorem 3. Decidability: The DDL-based rule conflict detection is decidable.

The above-mentioned rules can solve the rule conflict detection problem, for the problem can be converted into the satisfiability and consistency reasoning problems in the DDL.

Among these rules, ⊓, ∃ and ∀ rules can be completed in polynomial time. Despite the uncertainty of its completion time, ⊔ rule is a complete binary tree in the worst-case scenario, which can be completed in a limited time. Action α rule contains replacement operations, which can also be completed in a limited time. The final judgement based on the extension set has two results $\perp$ and $\top$. Therefore, The DDL-based rule conflict detection is decidable.

5. EXPERIMENTAL VERIFICATION

5.1 Experimental environment

The DDL-based reasoning of RAR conflicts between fine-grained resource attributes was verified through experiments on rule inference and conflict detection. As shown in Figure 3, the experimental model contains three main parts: an XACML-DDL converter, a rule analyzer, and an output generator.

The XACML-DDL converter is responsible for converting XACML rules into DDL semantics. During the working, the converter firstly traverses the element nodes in each rule, and loads the information of the corresponding rule. Then, the DDL-Converting is called to convert the information into a rule described in DDL language. Based on the instance and concept sub-models in the PIP ontology library, the rule analyzer performs logic reasoning on the rules described in DDL language, and feedbacks the reasoning results and anomaly handling to the output generator.

The rule inference components include attribute matching (Comparison), satisfiability verification (Verification) and conflicting rule checking (Conflict Checking). The rule analyzer provides an inference engine interface, allowing different DL inference engines (e.g. Pellet, Fact++, and Racer) [21-23] to participate in the rule inference of description logic. During Tableau-based rule inference, the inference engine is supported by the data from the ontology library provided by the decision-maker. The rules being inferred and their inference results are forwarded by the rule analyzer to the output generator for further processing. The output generator aggregates the abnormal rules into a rule set and feeds it back to the rule administrator, who will handle the abnormal rules.

The experimental data consists of two parts: Continue dataset and its expansion. Continue dataset was adopted by the XACML rule analyzer Margrave [24, 25], which is based on binary decision diagram (BDD). This dataset was selected to validate the XACML-based DL inference engine. The Continue dataset contains various complex and available XACML rules, which control the access of qualified users to article resources. A total of 26 rule files are provided in the dataset, including 86 RARs and 37 kinds of elements about user and article attributes. There are five description structures for identities and roles: *pc-member*, *pc-chair*, *subreviewer*, *editor* and *admin*. As shown in Table 1, the inheritance relationship in the dataset has a maximum of four layers: $pcMember \prec pcChair \prec subReviewer \prec editer \prec admin$.

The Continue dataset was also expanded, according to the situation in lightweight applications of attribute set and rule set in the IoT. The attribute descriptions of resources were refined, and the descriptions of IoT resource attributes were added to the original dataset. The expansion process is detailed in the next subsection.

Four layers of inheritance relationships may exist, depending on the identities and roles. The number of service attributes in the rules of the dataset and the number of rules is denoted as $\#\sum atbs_{rule}$ and $\#\sum rules_{pol}$, respectively. Different layers of inheritance may have the same RAR. The greater the number of inheritance layers, the more the attribute values in the rules. The experiments were carried out on Intel Pentium 4 2.4GHz CPU, 2GB memory, Windows XP SP3, and Java Runtime Environment 1.6.

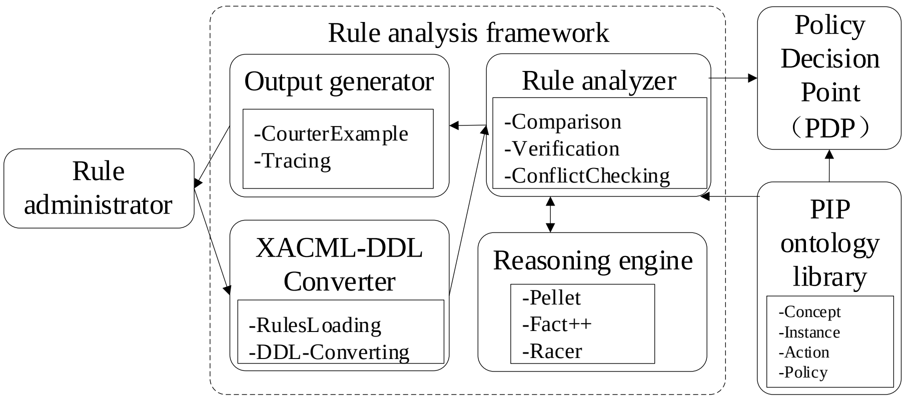

Figure 3. Analysis framework of rule inference experiments

Table 1. Inheritance relationships between attributes

Inheritance relationship	**CIR**	$\#\sum atbs_{rule}, \#\sum rules_{pol}$
One-layer inheritance	(*admin*, *editor*)	9, 24
Two-layer inheritance	(*editor*, *subreviewer*) (*admin*, *subreviewer*)	16, 33
Three-layer inheritance	(*subreviewer*, *pc-chair*) (*editor*, *pc-chair*) (*admin*, *pc-chair*)	28, 42
Four-layer inheritance	(*pc-chair*, *pc-member*) (*subviewer*, *pc-member*) (*editor*, *pc-member*) (*admin*, *pc-member*)	31, 69

Table 2. Time overheads of inference engines on authorization reasoning (unit: second)

Inheritance relationship	Pellet		Racer		Fact++		Margrave	
	Loading	Verify	Loading	Verify	Loading	Verify	Loading	Verify
One-layer inheritance	0.715	0.582	0.746	0.631	0.683	0.660	0.915	0.014
Two-layer inheritance	0.752	0.577	0.781	0.659	0.714	0.689	1.298	0.019
Three-layer inheritance	0.929	0.625	0.912	0.647	1.057	0.729	1.553	0.026
Four-layer inheritance	1.602	0.697	1.419	0.720	1.377	0.741	2.084	0.028

5.2 Performance evaluation

Rules with different inheritance layers were selected from the dataset to measure the time overhead of loading and reasoning for Continue dataset rules on the experimental platform. The time overhead on the platform is made up of the rule loading time (Loading) and rule verification time (Verify). The former refers to the time to traverse the elements in XACML rules and convert them into DDL logic description. The latter refers to the time to logically reason about DDL formal models based on the PIP.

The time overheads of four inference engines, namely, Pellet, Racer, Fact++ and Margrave, were compared. The first three inference engines are grounded on DDL formal descriptions, and the formal description and reasoning of the last engine are based on the BDD.

Firstly, a rule instance needs to be converted into a formal DDL description. Take a one-layer inheritance rule in Continue dataset for example. If the subject is an *admin* or *editor*, then the *meetingflag* of the conference turntable can be modified. The assertion P of the rule can be converted into a DDL formal description. In the sub-instance model, the instances and actions satisfy: $P \equiv (\exists \mathcal{M}_{Instance}^{Admin} \sqcup \mathcal{M}_{Instance}^{Editor}) \sqcap \exists \mathcal{M}_{Action}^{Write} \sqcap \exists \mathcal{M}_{Instance}^{MeetingFlag}$. Then, the satisfiability of the DDL formal description can be verified by the inference engine.

The time overheads of each inference engine on inheritance rules of different layers were recorded on the experimental platform (Table 2).

As shown in Table 2, the DL-based inference engines differed slightly in time overhead in the rule loading phase. Their time overheads were basically 1s, which is smaller than the BDD-based Margrave. The superiority of DL-based inference engines comes from the convenient conversion of XACML resource attributes into DL descriptions. Compared with Margrave, the DL-based inference engines are strong and efficient in formal expression of subjects, resources and RAR ontologies in the XML format.

In the verification phase, Margrave had an obvious advantage, as its time overheads were basically on the order of 10ms. This inference engine has a high efficiency in assertion verification, due to the rule inference based on the BDD. As for the three DL-based inference engines, the time overhead in rule inference and verification basically stabilized within 1s. The mean time overhead of Pellet stood at 0.6s, and that of Racer and Fact++ fell between 0.6s and 0.8s. Considering the actual application environment, the time overheads of DL-based inference engines, coupled with XACML conversion mechanism, are acceptable in the verification phase.

The Continue dataset offers a limited number of attributes and rules, which are insufficient to effectively simulate the application environment of the access authorization for cloud resources. Therefore, the dataset was expanded from two aspects, aiming to disclose the effects of the massive resources and RARs in cloud environment on the time overhead in DDL reasoning.

(1) Expanding attribute values without changing rule structure

Each attribute description in a rule was expanded by adding new attribute values. First, a similar attribute list $\{v_1, v_1, \cdots, v_{limit}\}$ was prepared for each attribute description in the rule, where limit is the threshold for the scale of attribute expansion. Next, each element node in the rule is traversed. If the original attribute value v was detected, it would be replaced by a random attribute in the similar attribute list. The expansion was terminated once all attribute values were replaced. The hierarchy of attributes was not changed through the expansion.

(2) Expanding rule set

The original rule set was expanded to increase the number of rules available. Specifically, a new reference node was added to the root node rule file (RPSlist.xml) in the Continue dataset. Under the node, new rules were generated by the XACML rule generator [26]. During the generation, the number of generated rules was controlled by configuring the following parameters: the maximum depth of rule tree (maxDepth), the maximum number of attributes (maxAttributePerCategory), the maximum attribute value (maxValuesPerAttribute), and the maximum number of rules (maxChildren). The attributes in the expanded rules must be those s that already exist in the Continue dataset.

The number of attributes and rules in the expanded Continue dataset could be controlled according to the experimental needs.

To verify the time overheads in conflict detection in Table 2, three experiments were designed to reveal (1) the effects of growing number of rules on the time overheads in detecting a single rule conflict, (2) the time overheads in detecting multiple rule conflicts at a fixed number of rules, and (3) the effects of the number of attributes on the efficiency of conflict detection.

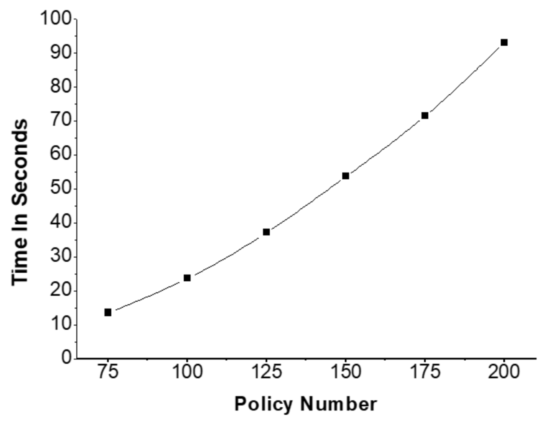

Figure 4. Effects of the number of rules on time overhead in detecting a single conflict

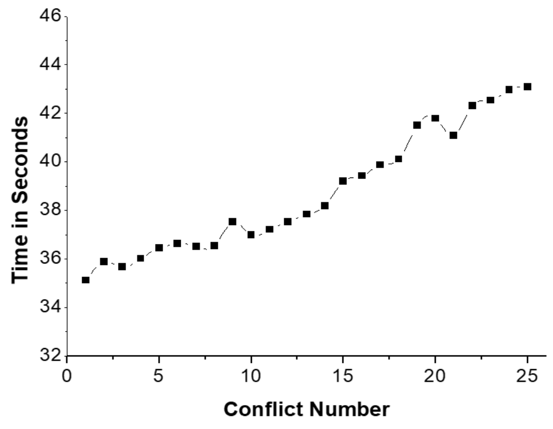

Figure 5. Effects of the number of rules on time overhead in detecting multiple conflicts

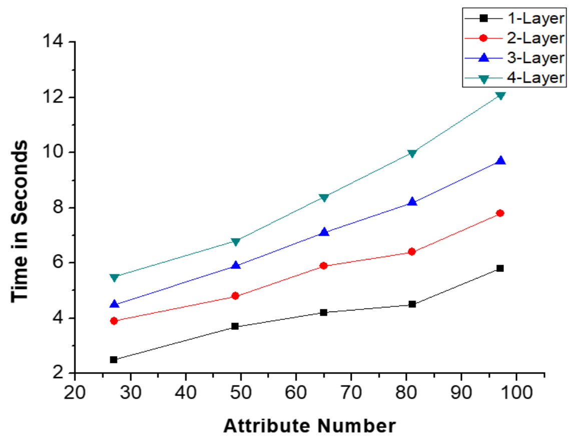

Figure 6. Effects of the number of attributes on the efficiency of conflict detection

In Experiment 1, six rule sets containing rule conflicts were selected from the expanded Continue dataset. The six sets include 50, 75, 100, 125, 175, and 200 rules, respectively. The algorithms in Subsection 4.2 were applied to detect the single rule conflict in these sets. The time overheads in conflict detection are recorded as Figure 4.

In Experiment 2, the number of rules in the rule set was controlled at 126. The proposed algorithms were adopted to detect the multiple rule conflicts. The time overheads in conflict detection are recorded as Figure 5.

The experimental results demonstrate that the efficiency of the conflict detection algorithms depends on the number of rules and conflicts in the rule set. As shown in Figure 4, the time overhead in detecting a single conflict increased linearly with the number of conflicts in the rule set. As shown in Figure 5, the time overhead in detecting multiple conflicts also increased linearly with the number of conflicts, except slight fluctuations at some nodes. For example, the time overhead in detecting 9 conflicting nodes differed from that in detecting 19 nodes. However, the fluctuation of time overhead was within 2s, and the time overheads for neighboring nodes exhibited no significant changes. Hence, the fluctuations can be neglected in actual applications. To sum up, the efficiency of conflict detection is basically linearly correlated with the number of conflicts. The proposed algorithms could detect the conflicts in the expanded Continue dataset within polynomial time, which satisfies the description in Theorem 3.

In Experiment 3, the Continue dataset was expanded by increasing attribute values without changing rule structure. Five rule sets with the same number of rules (175) were selected for the experiment. The number of attributes in the five sets was 27, 49, 65, 81 and 107, respectively. According to the hierarchical inheritance relationship among the policies, the data in five data sets are classified as four data groups. The proposed algorithms were adopted to detect the rule conflicts in these four data groups. The time overheads in conflict detection are recorded in Figure 6.

The four rule sets, which have different layers of inheritance, performed differently with the changing number of attributes. With the growing number of attributes, the time overheads of all four sets in conflict detection increased. The reason is that the addition of attributes increases the traversal time of all resource attribute nodes. The time overheads increased linearly with the number of attributes, which are acceptable on the expanded Continue dataset.

Owing to the difference in inheritance relationship, the four rule sets also differed in the time overhead under the same number of attributes. The difference is attributable to the following factors: a lower rule set has a smaller time overhead, because it contains fewer rules and the CIRs between its concepts could be computed in a shorter time. With the growth in the number of attributes, the time overhead difference between the rule sets continued to widen. This means, when the number and structure of rules remain the same, the number of attributes is the main influencing factor of the time overhead in conflict detection.

6. CONCLUSIONS

The XACML, as an attribute-based ACR description language, is suitable for authorizing access to resources in an open environment. Focusing on the possible RAR conflicts, this paper explores the causes of rule conflicts, and formally describes resource attributes with the DDL. Next, problems like attribute consistency and rule satisfiability were examined by setting up concept, instance and action knowledge bases. Then, a conflict detection algorithm was designed to identify the rule conflicts arising from the hierarchy of resource attributes. The algorithm relies on Tableau rules to detect rule conflicts in the light of satisfiability. The reliability and decidability of the algorithm were fully demonstrated. After that, three experiments were carried out to verify the feasibility of DDL-based authorization and reasoning, and to analyze the actual time overheads in loading and verification phases on expanded Continue dataset. The efficiency of the proposed algorithm was analyzed from three aspects: the number of rules, the number of conflicts, and the hierarchy of attributes. The results show that the DL-based inference is decidable in polynomial time.

REFERENCES

[1] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

[2] Atlam, H.F., Alassafi, M.O., Alenezi, A., Walters, R.J., Wills, G.B. (2018). XACML for building access control policies in internet of things. In IoTBDS, pp. 253-260. http://doi.org/10.5220/0006725102530260

[3] Ayache, M., Erradi, M., Khoumsi, A., Freisleben, B. (2016). Analysis and verification of XACML policies in a medical cloud environment. Scalable Computing:

Practice and Experience, 17(3): 189-206. http://doi.org/10.12694/scpe.v17i3.1180

[4] Kanwal, T., Jabbar, A.A., Anjum, A., Malik, S.U., Khan, A., Ahmad, N. (2019). Privacy-aware relationship semantics–based XACML access control model for electronic health records in hybrid cloud. International Journal of Distributed Sensor Networks, 15(6): 97-114. http://doi.org/10.1177/1550147719846050

[5] Charaf, L.A., Alihamidi, I., Addaim, A., Abdessalam, A.I.T. (2020). A distributed XACML based access control architecture for IoT systems. In 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), pp. 1-5. http://doi.org/10.1109/IRASET48871.2020.9092276

[6] Damiano, D., Paolo, M., Laura, R. (2019) A blockchain based approach for the definition of auditable access control systems. Computers & Security, 84: 93-119. https://doi.org/10.1016/j.cose.2019.03.016

[7] Ma, M., Shi, G., Li, F. (2019) Privacy-Oriented blockchain-based distributed key management architecture for hierarchical access control in the IoT scenario. IEEE Access, 7: 34045-34059. https://doi.org/10.1109/ACCESS.2019.2904042

[8] Zyskind, G., Nathan, O. (2015). Decentralizing privacy: Using blockchain to protect personal data. In 2015 IEEE Security and Privacy Workshops, pp. 180-184. https://doi.org/10.1109/SPW.2015.27

[9] Sukhodolskiy, I., Zapechnikov, S. (2018). A blockchain-based access control system for cloud storage. In 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), pp. 1575-1578. https://doi.org/10.1109/EIConRus.2018.8317400.

[10] Wang, S., Zhang, Y., Zhang, Y. (2018). A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access, 6: 38437-38450. https://doi.org/10.1109/ACCESS.2018.2851611

[11] Turkmen, F., den Hartog, J., Ranise, S., Zannone, N. (2017). Formal analysis of XACML policies using SMT. Computers & Security, 66: 185-203. https://doi.org/10.1016/j.cose.2017.01.009

[12] Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E. (2018, May). An automated model-based test oracle for access control systems. In 2018 IEEE/ACM 13th International Workshop on Automation of Software Test (AST), pp. 2-8. https://doi.org/10.1145/3194733.3194743

[13] Hsaini, S., Azzouzi, S., Charaf, M.E.H. (2019). FSM modeling of testing security policies for MapReduce frameworks. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 1480-1485. https://doi.org/10.1109/CoDIT.2019.8820685

[14] Lonetti, F., Marchetti, E. (2018). On-line tracing of XACML-based policy coverage criteria. IET Software, 12(6): 480-488. https://doi.org/10.1049/iet-sen.2017.0351

[15] Xu, D., Wang, Z., Peng, S., Shen, N. (2016). Automated fault localization of XACML policies. In Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies, pp. 137-147. https://doi.org/10.1145/2914642.2914653

[16] Calabrò, A., Lonetti, F., Marchetti, E. (2017). Access control policy coverage assessment through monitoring. In International Conference on Computer Safety, Reliability, and Security, pp. 373-383. https://doi.org/10.1007/978-3-319-66284-8_31

[17] Mourad, A., Tout, H., Talhi, C., Otrok, H., Yahyaoui, H. (2016). From model-driven specification to design-level set-based analysis of XACML policies. Computers & Electrical Engineering, 52: 65-79. https://doi.org/10.1016/j.compeleceng.2015.09.021

[18] Mejri, M., Yahyaoui, H., Mourad, A., Chehab, M. (2020). A rewriting system for the assessment of XACML Policies Relationship. Computers & Security, 97: 101957. https://doi.org/10.1016/j.cose.2020.101957

[19] Chernov, A.V., Butakova, M.A., Kartashov, O.O., Alexandrov, A.A. (2019). Intelligent decision support by means of dynamic description logic. In 2019 XXII International Conference on Soft Computing and Measurements (SCM), pp. 138-141. https://doi.org/10.1109/SCM.2019.8903760

[20] Baader, F., Gil, O.F., Marantidis, P. (2018). Matching in the Description Logic FL0 with respect to General TBoxes. In LPAR, pp. 76-94. https://doi.org/10.29007/q74p

[21] Vijayalakshmi, K., Jayalakshmi, V. (2020). A priority-based approach for detection of anomalies in ABAC policies using clustering technique. In 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), pp. 897-903. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-000166

[22] Silvestre, D., Hespanha, J., Silvestre, C. (2020). Resilient desynchronization for decentralized medium access control. IEEE Control Systems Letters, 5(3): 803-808. https://doi.org/10.1109/LCSYS.2020.3005819

[23] Sabitha, S., Rajasree, M.S. (2017). Access control based privacy preserving secure data sharing with hidden access policies in cloud. Journal of Systems Architecture, 75: 50-58. https://doi.org/10.1016/j.sysarc.2017.03.002

[24] Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C. (2005). Verification and change-impact analysis of access-control policies. In Proceedings of the 27th International Conference on Software Engineering, pp. 196-205. https://doi.org/10.1109/ICSE.2005.1553562

[25] http://www.margrave-tool.org/v1+v2/margrave/versions/01-01/documentation/.

[26] Sanchez, M., López, G., Gómez-Skarmeta, A.F., Cánovas, Ó. (2006). using microsoft office infopath to generate XACML policies. In International Conference on E-Business and Telecommunication Networks, Springer, Berlin, Heidelberg, pp. 134-145. https://doi.org/10.1007/978-3-540-70760-8_11