Semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies

Hussein Jebbaoui a, Azzam Mourad a,⇑, Hadi Otrok b, Ramzi Haraty a

a Department of Computer Science and Mathematics, Lebanese American University, Lebanon
b Department of Computer Engineering, Khalifa University of Science, Technology & Research, United Arab Emirates

ARTICLE INFO

Article history:
Received 15 May 2014
Received in revised form 4 October 2014
Accepted 16 December 2014
Available online 16 February 2015

Keywords:
Web services security
Access control
Policy analysis
Set theory
Semantics
XACML

ABSTRACT

XACML (eXtensible Access Control Markup Language) policies, which are widely adopted for defining and controlling dynamic access among Web/cloud services, are becoming more complex in order to handle the significant growth in communication and cooperation between individuals and composed services. However, the large size and complexity of these policies raise many concerns related to their correctness in terms of flaws, conflicts and redundancies presence. This paper addresses this problem through introducing a novel set and semantics based scheme that provides accurate and efficient analysis of XACML policies. First, our approach resolves the complexity of policies by elaborating an intermediate set-based representation to which the elements of XACML are automatically converted. Second, it allows to detect flaws, conflicts and redundancies between rules by offering new mechanisms to analyze the meaning of policy rules through semantics verification by inference rule structure and deductive logic. All the approach components and algorithms realizing the proposed analysis semantics have been implemented in one development framework. Experiments carried out on synthetic and real-life XACML policies explore the relevance of our analysis algorithms with acceptable overhead. Please visit http://www.azzammourad.org/#projects to download the framework.

1. Introduction

The heavy reliance on Web services as one of the primary methods for data exchange between partners and distributed systems still faces the risk of exploitation as a result of their infinite accessibility over the Internet [1,2]. In addition, services with critical data such as banking and other financial businesses are emerging, which increase security challenges [3]. In this regard, policy-based computing [4–6] is taking an increasing role in governing the systematic interaction among distributed services. Particularly, access control is the most challenging aspect of Web service security to determine which partner can access which service [7]. Currently, an increasing trend is to declare policies in a standardized specification language such as XACML, the OASIS standard eXtensible Access Control Markup Language [8]. Many vendors are adopting XACML for controlling access to their services.

Before stating the addressed problems and contributions of our work, we depict in the sequel a brief introduction about XACML [8], which has a policy structure divided into three layers. The top layer consists of a policy set, the middle layer consists of policies and the lower layer consists of rules. Each layer contains a target element which is used to define the
subjects, resources and actions. The policy set contains a set of policies, a set of obligations and a policy combining algorithm used to break the tie between its policies. Each policy has a set of rules, a set of obligations and a rule combining algorithm used to break the tie between its rules. A rule consists of a set of conditions and a rule effect. The obligations at the policy set and policy level are carried out when the final decision is reached to either permit or deny. The illustrative policy set example in Fig. 1, which will be used and explained in the case study (Section 5), depicts the policy structure.

Nowadays, mid and large size online systems may embed several distributed services heavily interacting and composed together to provide features satisfying the clients’ needs. This may require policies with hundreds and even thousands of rules to control access and enforce business behaviors. As a result, policies used as means of protection can be a source of weaknesses due to the presence of flaws and conflicts between their rules. For instance, considering the example in Fig. 1, rules R3 and R4 lead to an access flaw because both rules have no targets, both rules have the same effect Permit, R3 precedes in order R4 and R4 is more restricted than R3. With the current XACML decision mechanism, the generic rule R3 will always take precedence and be evaluated before the restricted rule R4. Therefore the response will always be given by R3 that grants access to any subject, while R4 that limits the access to subject Joe will be disregarded. In this context, the true objective of access control is to give higher priority to more restricted rules. Current XACML tools give major role to security administrators to resolve some tie/conflict decisions through policies/rules modifications and/or combining algorithms (e.g. Permit – overrides and First – Applicable). Although manual corrections may seem practical for small size policies, it is doubtful if not impossible for large ones within the complex structure of XACML. The problem grows more when integrating and composing different policies [9,2,7,10,6,5], where contradictions between combining algorithms are apparent. In this regard, some approaches have been proposed addressing XACML policy composition and analysis [11–18]. However, these propositions did not address the presence of access flaws, conflicts and redundancies between policies, and did not consider the logical meaning of rules that reflect the objectives of a policy.

In this paper, we tackle the aforementioned problems by elaborating a set-based scheme that provides formal specification of policies and semantics-based detection built on top of it to efficiently perform analysis tasks. The main contributions of this paper are two folds: (1) Addressing the complex constructs of XACML through an abstract set-based syntax (SBA-XACML), while maintaining a similar policy structure that covers all its elements and sub elements and (2) offering novel detection mechanisms that analyze the meaning of policy rules through semantics verification by inference rule structure and deductive logic. All the approach components and algorithms have been implemented in one development framework that accepts XACML policies as inputs, converts them automatically to SBA-XACML constructs, and produces a list of access flaws, conflicts and redundancies between rules. The provided experiments conducted on real-life and synthetic XACML policies explore the relevance and efficiency of our analysis approach with acceptable overhead.

The rest of the paper is organized as follows. Section 2 covers for the approach overview and architecture. Section 3 presents the semantics rules for policy and rule analysis. Section 4 illustrates the analysis algorithms. Section 5 depicts the case study and semantics-based detection. Section 6 focuses on the experiments and performance analysis. Section 7 summarizes the related work. Finally, Section 8 presents the conclusion.

2. Approach Overview

The overall architecture of our approach is illustrated in Fig. 2 with all its components, i.e. SBA-XACML Language, Compiler and Analysis Module. Using the framework, the user can analyse the policies for access flaws, conflicts and redundancies and get the corresponding analysis report using the module embedding the analysis algorithms.

2.1. SBA-XACML Language and Compiler

SBA-XACML is a set-based language composed of all the elements and constructs needed for the specification of XACML based policy. Please refer to [19] for the complete definition and syntax of SBA-XACML elements and attributes. Its compiler includes XACML parser and converter to SBA-XACML. It takes XACML policy set as inputs, parses their XACML elements and generates SBA-XACML constructs according to the language syntax and structure. In the sequel, we present a brief summary about its constructs that are needed in this paper. SBA-XACML based policy, referred to as a policy set PS, is ordered into 3 levels: PolicySet, Policy, and Rule. Every element can contain a Target. PolicySet element contains other PolicySet(s) and/or Policy(ie(s). Policy contains Rule(s).

A target TR is an objective and is mapped to SBA-XACML within the context of rule, policy and policy set according to the following syntax:

\[
TR = \{S, R, A\}
\]

where S is a set of subjects, R is a set of resources and A is a set of actions.

PS may contain other policy sets, policies or both. It can also be referenced by other policy sets. It is mapped to SBA-XACML according to the following syntax:

\[
PS :: = (ID, SP, PR, PCA, IPS, OBLs, TR)
\]
where $ID$ is the policy set id, $SP$ is the set of policies that belongs to policy set $PS$. $PR$ is the precedence order of policies that belongs to $PS$. $PCA$ is the policy combining algorithm, $IPS$ is the policies or policy set that are referenced by $PS$. $OBLs$ is the set of obligations and $TR$ is the target.

A policy $P$ contains a set of rules, rule combining algorithm, target and obligations. It is mapped to SBA-XACML according to the following syntax:

$$P ::= (ID, SR, PR, RCA, OBLs, TR)$$

where $ID$ is the policy id, $SR$ is the set of rules that belongs to policy $P$. $PR$ is the precedence order of rules that belongs to $P$. $RCA$ is the rule combining algorithm, $OBLs$ is the set of obligations and $TR$ is the target.

A rule $R$ is the most elementary element of a policy. A rule contains rule conditions, target and rule effect. It is mapped to SBA-XACML according to the following syntax:

```xml
<Policy PolicyId="PI" RuleCombiningAlgId="rule-combining-algorithm:deny-overrides"> 
  <Condition>
    <Apply FunctionId="function:and"> 
      <Apply FunctionId="function:string-equal"> 
        <ResourceAttributeDesignator AttributeId="resource:resource-id" 
          DataType="xml:string" />
        <ApplyValue DataType="xml:string">Bob</ApplyValue>
      </Apply>
    </Apply>
    <Apply FunctionId="function:string-equal"> 
      <ResourceAttributeDesignator AttributeId="subject:subject-id" 
        DataType="xml:string" />
      <ApplyValue DataType="xml:string">Joe</ApplyValue>
    </Apply>
  </Condition>
  <Rule Effect="Deny" RuleId="R2" />
</Policy>
```
where ID is the rule id, RC is the set of rule conditions, TR is the target and RE is the rule effect.

A rule condition RC is a boolean function over subjects, resources, actions or functions of attributes. It is mapped to SBA-XACML within the context of a rule according to the following syntax:

RC = \{Apply, function, parameters\}

where \textit{Apply, function} is the function used in evaluating the elements and \textit{parameters} are the input to the function being applied.

2.2. Policy analysis module

This module allows to analyse policies for detecting access control flaws, conflicts and redundancies between rules. It is composed of policy-level and rule-level analysis algorithms that realize the elaborated analysis semantics presented in Section 3. The policy-level algorithm is responsible for analysing policies and triggers the rule-level one in order to analyze the rules in each policy. The analysis module works effectively if scheduled as a trigger on the repository to run whenever any modification is performed on policies. It can be scheduled to run in parallel with policy evaluation as well. It accepts a policy set as input and generates an analysis report.

3. Semantics-based analysis

The structural operational semantics used in this paper is an approach proposed to give logical means in defining operational semantics [20,21]. It defines the behavior of a process in terms of the behavior of its parts. Computation is represented by means of deductive logic that turn the abstract machine into a system of logical inferences. This allows to apply formal analysis on the behavior of processes. The behavior of a process is defined in terms of a set of transition relations. Such specifications take the form of inference rules. Definitions are given by inference rules, which consist of a conclusion that follows from a set of premises, possibly under control of some conditions. An inference rule has a general form consisting of the premises listed above a horizontal line, the conclusion below, and the condition, if present, to the right [21].

In this section, we present the formal semantics of SBA-XACML policy analysis following the inference rule structure and deductive logic. Given a policy set PS, the analysis report R is derived from the evaluation \( PA \) of all premises combined between each other using designated operators \( op \) as follows:

\[
(\text{premise}_1) \; \text{op} \; (\text{premise}_2) \; \text{op} \; \ldots \; \text{op} \; (\text{premise}_n) \\
(PS) \rightarrow R
\]

Throughout the rest of the paper, please note the difference between a semantic rule that expresses the analysis at a particular level, and a policy rule which is a construct in SBA-XACML. All the semantics rules follow the bottom up structure, where all the common ones are presented first, then followed by the rule level, policy level and policy set level ones.
3.1. Subset and intersection function

Rule R1 has subject set S1, resource set R1 and action set A1. Rule R2 has subject set S2, resource set R2 and action set A2. Rules 1 and 2 in Table 1 describe the different cases of subset rules. In Rule 1, a target TR1 is a subset of target TR2 if subject set S1 is a subset of subject set S2, resource set R1 is a subset of resource set R2, and action set A1 is a subset of action set A2. In Rule 2, a target TR1 is not a subset of target TR2 if subject set S1 is not a subset of subject set S2, or resource set R1 is not a subset of resource set R2, or action set A1 is not a subset of action set A2.

Rules 3 and 4 in Table 2 describe the different cases of intersection rules. In Rule 3, two targets TR1 and TR2 intersect if subject set S1 and subject set S2 share common elements, resource set R1 and resource set R2 share common elements, and action set A1 and action set A2 share common elements. In Rule 4, two targets TR1 and TR2 do not intersect if subject set S1 and subject set S2 share no common elements, resource set R1 and target resource set R2 share no common elements, or action set A1 and action set A2 share no common elements.

3.2. Flaw detection

Table 3 presents the analysis semantics rules for detecting access flaws in SBA-XACML policy set. Two rules R1 and R2 cause access flaw if: (1) Both share the same effect (i.e. access decision) over the same set of subjects, resources and actions, (2) R2 is more restricted than R1 and (3) R1 takes a precedence order over R2, i.e. the decision is based on the evaluation of R1 while R2 is ignored. In other words, the access decision is taken based on the more general rule R1 evaluated first, while the more restricted R2 is ignored. Accordingly, semantics Rules 5–8 realize the aforementioned logic and describe the different access flaw analysis cases for a policy set PS. It is worth noting that such flaws cannot be resolved with the current combining algorithms of XACML.

In Rule 5, two rules R1 and R2 return flaw if R2 target TR2 is a subset of rule R1 target TR1, rule condition RC2 is a subset of R1 rule condition RC1, and both R1 and R2 have the same rule effect, i.e. RE1 is equal to RE2. In Rule 6, for every pair of rules R1 and R2 in policy P such that R1 and R2 are appended to the Flaw Set FS if R1 and R2 evaluate to flaw. In Rule 7, given a pair of policies P1, P2, and rule condition P and P2 are appended to the Flaw Set FS if the rule combining of P1 is equal to the rule combining of P2, the targets of P1 and P2 intersect, and there exists R1 in P1 and R2 in P2 such that R1 and R2 evaluate to flaw and appended to the Flaw Set FS. In Rule 8, given policy set PS, the Flaw Set FS is the union of all flaws between any two flawed rules R1 and R2 in one policy and between two flawed rules R1 and R2 from every two flawed policies P1 and P2.

3.3. Redundancy detection

Table 4 presents the analysis semantics rules for detecting redundancies in SBA-XACML policy set. Two rules R1 and R2 are considered redundant if both have the same effects (i.e. access decision) over the same set of subjects, resources and actions. Accordingly, semantics Rules 9–12 realize the aforementioned logic and describe the different conflict analysis cases for a policy set PS.

In Rule 9, two rules R1 and R2 are redundant if R1 target TR1 and R2 target TR2 intersect, rule condition RC1 and rule R2 rule condition RC2 intersects, and both rules R1 and R2 have the same rule effect, i.e. RE1 is equal to RE2. In Rule 10, for every pair of rules R1 and R2 in policy P such that R1 and R2 are appended to the Redundant Set RS if R1 and R2 are redundant. In Rule 12, given a pair of policies P1, P2 in policy set PS, P1 and P2 are appended to the Redundant Set RS if the rule combining of P1 is equal to the rule combining of P2, the targets of P1 and P2 intersect, and there exists R1 in P1 and R2 in P2 such that R1 and R2 are redundant and appended to the Redundant Set RS. In Rule 12, given policy set PS, the Redundant Set RS is the union of all redundancies between any two redundant rules R1 and R2 in one policy and between two redundant rules R1 and R2 from every two redundant policies P1 and P2.

**Table 1**

<table>
<thead>
<tr>
<th>Subset function semantics rules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S1 \subseteq S2) \land (R1 \subseteq R2) \land (A1 \subseteq A2))</td>
</tr>
<tr>
<td>((\langle TR1, TR2 \rangle) \leftarrow \text{subset} \text{ True} )</td>
</tr>
<tr>
<td>((S1 \not\subseteq S2) \lor (R1 \not\subseteq R2) \lor (A1 \not\subseteq A2))</td>
</tr>
<tr>
<td>((\langle TR1, TR2 \rangle) \leftarrow \text{subset} \text{ False} )</td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>Intersection function semantics rules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S1 \cap S2 \neq \emptyset) \land ((R1 \cap R2 \neq \emptyset)) \land ((A1 \cap A2 \neq \emptyset)))</td>
</tr>
<tr>
<td>((\langle TR1, TR2 \rangle) \leftarrow \text{intersect} \text{ True} )</td>
</tr>
<tr>
<td>((S1 \cap S2 = \emptyset) \lor ((R1 \cap R2 = \emptyset)) \lor ((A1 \cap A2 = \emptyset)))</td>
</tr>
<tr>
<td>((\langle TR1, TR2 \rangle) \leftarrow \text{intersect} \text{ False} )</td>
</tr>
</tbody>
</table>
### Table 3
Rules of access flaw detection semantics.

$$\left( (TR2, TR1) \vdash_{\inter} True \right) \land (RC2 \subseteq RC1) \land (RE1 = RE2)$$  \hspace{1cm} (Rule 5)

$$\left( (R1, R2) \vdash_{\text{RA}} \text{Flaw}_{R1,R2} \right)$$  \hspace{1cm} (Rule 6)

$$\left( \forall R1, R2 \in SR, FS \rightarrow FS \cup \left( (R1, R2) \vdash_{\text{RA}} \text{Flaw}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 7)

$$\left( (RC A P1 = RCA P2) \land (TR1, TR2) \vdash_{\inter} true \right) \land \left( \forall R1 \in SR1, R2 \in SR2, FS \rightarrow FS \cup \left( (R1, R2) \vdash_{\text{RA}} \text{Flaw}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 8)

### Table 4
Rules of redundancy detection semantics.

$$\left( (TR2, TR1) \vdash_{\inter} True \right) \land (RC2 \cap RC1 \neq \emptyset) \land (RE1 = RE2)$$  \hspace{1cm} (Rule 9)

$$\left( (R1, R2) \vdash_{\text{RA}} \text{Redundant}_{R1,R2} \right)$$  \hspace{1cm} (Rule 10)

$$\left( \forall R1, R2 \in SR, FS \rightarrow FS \cup \left( (R1, R2) \vdash_{\text{RA}} \text{Redundant}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 11)

$$\left( (RC A P1 = RCA P2) \land (TR1, TR2) \vdash_{\inter} true \right) \land \left( \forall R1 \in SR1, R2 \in SR2, FS \rightarrow FS \cup \left( (R1, R2) \vdash_{\text{RA}} \text{Redundant}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 12)

### Table 5
Rules of conflict detection semantics.

$$\left( (TR1, TR2) \vdash_{\inter} True \right) \land (RC1 \cap RC2 \neq \emptyset) \land (RE1 = RE2)$$  \hspace{1cm} (Rule 13)

$$\left( (R1, R2) \vdash_{\text{CA}} \text{Conflict}_{R1,R2} \right)$$  \hspace{1cm} (Rule 14)

$$\left( \forall R1, R2 \in SR, FS \rightarrow RS \cup \left( (R1, R2) \vdash_{\text{CA}} \text{Conflict}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 15)

$$\left( (RC A P1 = RCA P2) \land (TR1, TR2) \vdash_{\inter} true \right) \land \left( \forall R1 \in SR1, R2 \in SR2, FS \rightarrow RS \cup \left( (R1, R2) \vdash_{\text{CA}} \text{Conflict}_{R1,R2} \right) \right)$$  \hspace{1cm} (Rule 16)

3.4. Conflict detection

Table 5 presents the analysis semantics rules for detecting conflicts in SBA-XACML policy set. Two rules R1 and R2 cause conflict if both have opposite effects (i.e. access decision) over the same set of subjects, resources and actions. Accordingly, semantics Rules 13–16 realize the aforementioned logic and describe the different conflict analysis cases for a policy set PS. Once such conflict is detected, the current combining algorithms of XACML can be used to resolve it.

In Rule 14, two rules R1 and R2 conflict if R1 target TR1 and R2 target TR2 intersect, rule condition RC1 and R2 rule condition RC2 intersects and R1 with effect RE1 is the opposite of R2 with effect RE2. In Rule 14, for every pair of rules R1 and R2 in policy P such that R1 and R2 are appended to the Conflict Set CS if R1 and R2 conflict. In Rule 15, given a pair of policies P1, P2 in policy set PS, P1 and P2 are appended to the Conflict Set CS if the rule combining of P1 is equal to the rule combining of P2, the targets of P1 and P2 intersect, and there exists R1 in P1 and R2 in P2 such that R1 and R2 conflicted and appended to the Conflict Set CS. In Rule 16, given policy set PS, the Conflict Set CS is the union of all conflicts between any two conflicting rules R1 and R2 in one policy and between two conflicting rules R1 and R2 from every two conflicting policies P1 and P2.
4. Policy analysis algorithms

In this section, we present the algorithms realizing the SBA-XACML policy analysis semantics. The analysis module is divided into three algorithms: (1) the rule analysis algorithm is presented in Algorithm 1, (2) the policy analysis algorithm in Algorithm 2 and (3) policy set analysis algorithm in Algorithm 3.

4.1. Rule analysis algorithm

The Rule Analysis Algorithm is presented in Algorithm 1. It takes two rules \( R_1 \) and \( R_2 \) as input and compares their targets, rule conditions and rule effects to determine if there exists any flaw, conflict or redundancy between the two rules. It returns the proper response to the Policy Analysis Algorithm in Algorithm 2. If the target of rule \( R_2 \) is a subset of the target of rule \( R_1 \), the rule condition set of \( R_2 \) is a subset of the rule condition set of \( R_1 \) and \( R_1 \) and \( R_2 \) have the same effect and \( R_1 \) takes a precedent order over \( R_2 \), then the rule \( R_1 \) is considered as access control flaw. If the subject set of \( R_1 \) intersects with subject set of \( R_2 \), resource set of \( R_1 \) intersects with resource set of \( R_2 \) and action set of \( R_1 \) intersects with action set of \( R_2 \) and \( R_1 \) and \( R_2 \) have opposite effect, then \( R_1 \) conflicts with \( R_2 \). Otherwise, if \( R_1 \) and \( R_2 \) have the same effect then \( R_1 \) and \( R_2 \) are redundant. Empty set is returned if no issues were found between the two rules.

Algorithm 1. Rule_Analysis \((R_1, R_2)\)

```plaintext
Input: Two Rules \( R_1 \) with Target \( TR_1 = \{S_1, R_1, A_1\} \), rule condition \( RC_1 \), rule effect \( RE_1 \) and \( R_2 \) with Target \( TR_2 = \{S_2, R_2, A_2\} \), rule condition \( RC_2 \), rule effect \( RE_2 \)
Output: Rule analysis \( \in \{\text{Flaw, Conflict, Redundant or Null}\} \)
1: if \((S_2 \subseteq S_1) \land (R_2 \subseteq R_1) \land (A_2 \subseteq A_1)\) then
2: if \((RC_2 \subseteq RC_1)\) then
3: if \((RE_1 = RE_2)\) then
4: // R2 is a subset of R1
5: return "Flaw";
6: end if
7: end if
8: end if
9: if \(((S_1 \cap S_2) \neq \emptyset) \land ((R_1 \cap R_2) \neq \emptyset) \land ((A_1 \cap A_2) \neq \emptyset)\) then
10: if \((RC_1 \cap RC_2) \neq \emptyset\) then
11: if \((RE_1 \neq RE_2)\) then
12: return "Conflict";
13: else
14: return "Redundant";
15: end if
16: end if
17: end if
18: return;
```

4.2. Policy analysis algorithm

The policy analysis algorithm is presented in Algorithm 2. It takes two policies \( P_1 \) and \( P_2 \) as input and produces a set of all access flaws \( FS \), conflicts \( CS \) and redundancies \( RS \). The algorithm is composed of two parts. The first part of the algorithm checks for flaws, conflicts and redundancies within each policy. The second part of the algorithm checks for them between rules from different policies. The returned responses from the Rule Analysis calls are appended to the proper sets.

Algorithm 2. Policy_Analysis \((P_1, P_2)\)

```plaintext
Input: Policy \( P_1 \) with Target \( TR_1 = \{S_1, R_1, A_1\} \) and \( P_2 \) with Target \( TR_2 = \{S_2, R_2, A_2\} \)
Output: Flaw Set \( FS \), Conflict Set \( CS \) and Redundancy Set \( RS \)
1: //Check rules in each policy
2: for \( l := 1 \) to \( 2 \) do
3: for \( i := 1 \) to \( P_1 \text{Number of Rules} - 1 \) do
4: for \( j := 2 \) to \( P_2 \text{Number of Rules} \) do
(continued on next page)
```
if (RULE_ANALYSIS (Rli, Rlj) = "Flaw") then
7:     FS = FS \cup Flaw_{Rli, Rlj};
end if

if (RULE_ANALYSIS (Rli, Rlj) = "Redundant") then
9:     RS = RS \cup Redundant_{Rli, Rlj};
end if

if (RULE_ANALYSIS (Rli, Rlj) = "Conflict") then
11:    CS = CS \cup Conflict_{Rli, Rlj};
end if
end for

end for

end for

//Check rules in both P1 and P2
if (RCAP_{1} = RCAP_{2}) then
17: for l := 1 to P_{1. NumberofRules} do
18:     for m := 1 to P_{2. NumberofRules} do
19:         if (RULE_ANALYSIS(R_{l}, R_{m}) = "Flaw") then
20:             FS = FS \cup Flaw_{R_{l}, R_{m}};
21:         end if
22:         if (RULE_ANALYSIS(R_{l}, R_{m}) = "Conflict") then
23:             CS = CS \cup Conflict_{R_{l}, R_{m}};
24:         end if
25:         if (RULE_ANALYSIS(R_{l}, R_{m}) = "Redundant") then
26:             RS = RS \cup Redundant_{R_{l}, R_{m}};
27:         end if
28:     end for
29: end for
end if
return;

4.3. PolicySet Analysis algorithm

The PolicySet Analysis algorithm is presented in Algorithm 3. It takes a policy set PS as input and produces a report of all access flaws, conflicts and redundancies between policies and rules. It initializes global set FS, CS and RS and calls the Policy Analysis algorithm in Algorithm 2 on for checking and appending flaws, conflicts and redundancies at both policy and rule levels, in each policy and between every two policies.

Algorithm 3. PolicySet Analysis(PS)

Input: A Policy Set PS
Output: Analysis report
1: Global FS = \emptyset; RS = \emptyset; CS = \emptyset;
2: for l := 1 to PS_{NumberofPolicies} do
3:     for j := l + 1 to PS_{NumberofPolicies} do
4:         PA = POLICY ANALYSIS(P_{l}, P_{j});
5:     end for
6: end for
5. Case study: Semantics-based policy analysis

In this section, we present a case study illustrating the practicality of SBA-XACML policy analysis process through semantics rules. Listing 1 contains the generated SBA-XACML based policy corresponding to the XACML policy example presented in Fig. 1.

Line 1 is the policy set \( P \). The policy set \( ID \) is \( P.S1 \). It has two policies \( P1 \) and \( P2 \). \( P1 \) is ordered before \( P2 \). The policy combining algorithm is \( \text{permit} – \text{overrides} \). \( P1 \) has no reference to other policies. It has no obligations to perform and the target subjects, resources and actions are any. Line 2 is the policy \( P \). The policy \( ID \) is \( P1 \). It has two rules \( R1 \) and \( R2 \). \( R1 \) is ordered before \( R2 \). The rule combining algorithm is \( \text{deny} – \text{overrides} \). \( P1 \) has no obligations and no target. Line 3 is the rule \( R \). The rule \( ID \) is \( R1 \). \( R1 \) has a set of conditions. The conditions are: the subject \( ID \) must be equal to \( Bob \) and the resource \( ID \) must be equal to BankService/Withdraw. The target subjects, resources and actions are any. \( R1 \) has a \( \text{permit} \) effect. Line 4 is the rule \( R2 \). The rule \( ID \) is \( R2 \). \( R2 \) has no conditions. \( R2 \) has no target specified. \( R2 \) has a \( \text{deny} \) effect. Line 5 is the policy \( P2 \). The policy \( ID \) is \( P2 \). It has three rules \( R3 \), \( R4 \) and \( R5 \). The precedence order is \( R3 \), \( R4 \) then \( R5 \). The rule combining algorithm is \( \text{permit} – \text{overrides} \). \( P2 \) has no obligation to perform and the target elements are not defined. Line 6 is the rule \( R3 \). The rule \( ID \) is \( R3 \). \( R3 \) has one condition. The condition states that the resource \( ID \) must be equal to BankService/Deposit. The target subjects, resources and actions are not specified. \( R3 \) has a \( \text{permit} \) effect. Line 7 is the rule \( R4 \). The rule \( ID \) is \( R4 \). \( R4 \) has a set of conditions. The conditions are: the subject \( ID \) must be equal to \( Joe \) and the resource \( ID \) must be equal to

**SBA-XACML Policy for a Bank Service**

```xml
Listing 1. SBA-XACML policy for a bank service.
```

\[
\begin{align*}
&\text{SBA-XACML Policy for a Bank Service} \\
&[1]. P.S := <P.S1, P1, P2, P1>P2, (\text{Permit overrides}), \{\{\}, \{\}, \{\}, \{\}, \{\}, \{\}>
\end{align*}
\]

\[
\begin{align*}
&\text{Line 1 is the policy set } P.S. \text{ It has two policies } P1 \text{ and } P2. \text{ } P1 \text{ is ordered before } P2. \text{ The policy combining algorithm is permit – overrides. } P1 \text{ has no reference to other policies. It has no obligations to perform and the target subjects, resources and actions are any. Line 2 is the policy } P. \text{ The policy } ID \text{ is } P1. \text{ It has two rules } R1 \text{ and } R2. \text{ } R1 \text{ is ordered before } R2. \text{ The rule combining algorithm is deny – overrides. } P1 \text{ has no obligations and no target. Line 3 is the rule } R. \text{ The rule } ID \text{ is } R1. \text{ } R1 \text{ has a set of conditions. The conditions are: the subject } ID \text{ must be equal to } Bob \text{ and the resource } ID \text{ must be equal to BankService/Withdraw. The target subjects, resources and actions are any. } R1 \text{ has a permit effect. Line 4 is the rule } R2. \text{ The rule } ID \text{ is } R2. \text{ } R2 \text{ has no conditions. } R2 \text{ has no target specified. } R2 \text{ has a deny effect. Line 5 is the policy } P2. \text{ The policy } ID \text{ is } P2. \text{ It has three rules } R3, R4 \text{ and } R5. \text{ The precedence order is } R3, R4 \text{ then } R5. \text{ The rule combining algorithm is permit – overrides. } P2 \text{ has no obligation to perform and the target elements are not defined. Line 6 is the rule } R3. \text{ The rule } ID \text{ is } R3. \text{ } R3 \text{ has one condition. The condition states that the resource } ID \text{ must be equal to BankService/Deposit. The target subjects, resources and actions are not specified. } R3 \text{ has a permit effect. Line 7 is the rule } R4. \text{ The rule } ID \text{ is } R4. \text{ } R4 \text{ has a set of conditions. The conditions are: the subject } ID \text{ must be equal to } Joe \text{ and the resource } ID \text{ must be equal to}
\end{align*}
\]
BankService/Deposit. The target subjects, resources and actions are not specified. R4 has a permit effect. Line 8 is the rule R5. The rule ID is R5. R5 has a set of conditions. The conditions are: the subject ID must be equal to Joe and the resource ID must be equal to BankService/Deposit. The target subjects, resources and actions are not specified. R5 has a deny effect.

Based on the SBA-XACML policy analysis semantics in Section 3, the elaborated framework will analyze the based policy PS1 presented in Listing 1 for detecting access flaws, conflicts and redundancies. The result of the analysis shows that PS1 has FlawR1,3, R4, ConflictR1,2; R3, R5 and RedundantR1,3, R4. To avoid repetition and for space limitation, we will only present the access flaws detection and three different cases where flaws exist between two rules, no flaws between two rules and no flaws between two policies in PS1. The analysis for conflict and redundancy detection is performed in similar way. The analysis of each semantics rule in Fig. 3 is based on analyzing its premises, hence they should be read from bottom up, i.e at the level of policy sets, policies and then rules as follows:

1. The based policy is composed of a PolicySet PS1. PS1 contains a flaw because there exists a policy P2 in PS1 such that P2 contains a flaw between rules R3 and R4 as depicted in (3). Hence, based on the semantics (Rule 8) that applies in this case and combines the results of all the premises, the final response is the flaw set FS containing FlawR1,3, R4.
2. Policy P1 is composed of two rules R1 and R2, which cause no flaw by applying semantics (Rule 5). First, Premise1 evaluates to True because R1 and R2 have no targets defined, which means TR1 = \{S = Any, R = Any, A = Any\} and TR2 = \{S = Any, R = Any, A = Any\}. Second, R2 rule conditions RC2 is not a subset of R1 rule conditions RC1 since R2 has no condition, while R1 has two conditions that limit the access to resource equals to BankService/Deposit and subject equals to Joe. Hence, Premise2 evaluates to False. This fact denies the presence of a flaw, where the decision is based on the restricted rule that precedes the general one. Hence, the response is null and there is no need to continue checking the remaining premises.
3. Policy P2 is composed of three rules R3, R4 and R5. Rules R3 and R4 cause flaw by applying semantics (Rule 5). First, Premise1 evaluates to True because R3 and R4 have no targets defined, which means TR3 = \{S = Any, R = Any, A = Any\} and TR4 = \{S = Any, R = Any, A = Any\}. Second, R4 rule conditions RC4 is a subset of R3 rule conditions RC3 since both rules require the resource to be BankService/Deposit, while R4 limits the subject to Joe. Hence, Premise2 evaluates to True. This fact constitutes the first sign for a flaw, where the decision is based on the general rule that precedes the restricted one. Third, Premise3 evaluates to True because both rules R3 and R4 have the same effect Permit. All premises of semantics (Rule 5) evaluate to true, therefore policy P2 has access flaw between rules R3 and R4. The same analysis steps will be applied between R3 and R5 and between R4 and R5, without any existence of flaws. We did not include them to avoid repetition. Hence, the response is the flaw set FS containing FlawR1,3, R4.
4. Policies P1 and P2 do not cause flaw. By Applying semantics (Rule 7), Premise1 evaluates to False because the rule combining algorithm RCA1 of P1 is equal to deny – overrides, which is different than the rule combining algorithm RCA2 of P2, which is equal to permit – overrides. Hence, the response is null and there is no need to continue checking the remaining premises and analysing the rules of P1 and P2.

6. Discussion and experimental results

In this section, we examine the results of our experiments for analyzing policies for access flaws, conflicts and redundancies. The SBA-XACML framework is implemented in PHP. The experiments were carried out on a notebook running Windows XP SP3 with 3.50 GB of memory and dual core 2.8 GHz Intel processor. The tests were conducted on both real world and synthetic policies to show the scalability and performance whether small or large. The real policies utilized in the experiments are small and mid-sized ones ranging between 2 and 298 rules. The synthetic policies are small and large ranging between 400 and 4000 rules. The flawed, conflicted and redundant rules were injected at random with different rate from 1 to 5 per every 10 rules. This process has been repeated hundreds of times with both real and synthetic policies.

The first set of experiments has been performed to assess the detection rate of our approach. At each trial, extensive testing has been performed in order to make sure that the proposed detection mechanisms are able to successfully detect all the

![Policy Analysis Results Analysis](image-url)
injected flaws, conflicts and redundancies. The results of our experiments gave 100% detection rate. The second set of experiments were conducted to assess the performance and efficiency of the analysis process. The processing time is the average time calculated based on 100 K run per every policy set. Our analysis experiments were performed based on two scales, the normal case and the worst case. The normal case considers having policies injected with flaws, conflicts and redundancies up to 10%, while the worst case injected up to 50%. The analysis processing time is static and includes the conversion of the policy set from XACML to SBA-XACML, which is optional and executed only once when deploying policies.

Fig. 4a and b show the experimental results for analyzing real life policies utilizing the normal scale, with injecting flaws, conflicts and redundancies up to 10%. Policies with 4 rules require less than 40 m-s to complete the analysis process, while policies with 298 rules require on average about 24.2 s to complete.

Fig. 5a presents the statistics for analyzing synthetic policies utilizing two different scales, the normal and the worst. Fig. 5a shows the processing time when 1 out 10 rules/policies cause access flaws, conflicts or redundancies. At 400 rules, the analysis processing time consumes 5.54 s to complete. At 1200 rules, it consumes 44 s. And at 4000 rules, it consumes 692 s.

Fig. 5b shows the processing time when five out of ten rules/policies cause access flaws, conflicts or redundancies. The policies are designed in such a way that not only rules within each policy are verified, but also rules from different policies are verified as well due to similarities between policy targets and rule combining algorithms. The statistics show that policies with 400 rules can be analyzed with 29 s, 1200 rules with 352 s and 4000 rules with 7210 s. The worst case processing times seem reasonable considering the policy size and the percentage of flaws, conflicts and redundancies injected. However, such cases are very unlikely to incur in real world policies. The results of these experiments explore the efficiency in terms of performance for reasonable size policies. The addition overhead is affordable, even though such analysis is performed most of the time offline.

7. Related work

In this section, we provide an overview of the related work in the literature addressing XACML policy analysis, in addition to some policy evaluation approaches. In this regard, Kolovski et al. [11] proposed a formalization of XACML using description logics (DL), which are a decidable fragment of First-Order logic. They perform policy verification by using the existing DL verifiers. Their analysis service can discover redundancies at the rule level. However, they do not address access flaws and do not support multi-subject requests, complex attribute functions, rule Conditions and Only-One-Applicable combining algorithm.

Fisler et al. [13] proposed a suite called Margrave. It verifies whether an access control policy satisfies a given property and computes the semantic difference of two XACML policies. However, their proposal does not address policy analysis with respect to access flaws, and does not work on all types of XACML policies. Tschantz and Krishnamurthi [14] present a set of properties for examining the reasonability of access control policies under enlarged requests, policy growth, and policy decomposition. However, they do not address policy analysis with respect to access flaws. Mazzoleni et al. [15] proposed an authorization technique for distributed systems with policies from different parties. Their approach focuses on finding similarities between policies from different parties based on requests. This approach focuses on policy integration from different parties and do not address policy analysis for flaws.

Rao et al. [16] introduced an algebra for fine-grained integration that supports specification of a large variety of integration constraints. They introduced a notion of completeness and proved that their algebra is complete with respect to this notion. Their approach, however, does not cover rule conditions and obligations and focuses on integration between different parties, unlike ours which focuses on analyzing policy sets individually and after integration. Wijesekera and Jajodia [17] have proposed algebra for manipulating access control policies at a higher level, where the operations of the algebra are abstracted from their specification details. However, they do not address XACML and do not provide implementation for
their algebra. Bonatti et al. [18] introduced the concept of policy composition under constraints and proposed algebra for composing access control policies using a variable free authorization terms such as subject, object and action. However, this approach focuses on policy composition from distributed parties and do not target XACML.

In another context related to policy evaluation for efficient decision process, few approaches have been proposed such [22–25]. Based on the study of the current literature, it is trivial that both domains are still and will continue to be a challenging niche for researchers. To the best of our knowledge, none of the current approaches did address the presence of access flaws, conflicts and redundancies between policies, and did consider the logical meaning of rules that reflect the objectives of a policy. In this regard, our approach differs by providing a novel detection mechanisms that study the meaning of policy rules through semantics verification by inference rule structure and deductive logic.

8. Conclusion and future work

The proposed approach addressed the problem of flaws, conflicts and redundancies presence between the rules of large-size and complex XACML policies. In this context, the contribution of this work is the elaboration of a set-based scheme that provides formal specification of XACML policies and semantics-based detection built on top of it to efficiently perform analysis tasks. Our approach improves the related literature in two different aspects. First, it offers an abstract set-based language that addresses the complex constructs of XACML while maintaining similar policy structure that covers all its constructs. Second, it embeds a novel detection mechanism that analyzes the meaning of policy rules through semantics verification by inference rule structure and deductive logic. The aforementioned theoretical outcomes were realized by developing practical algorithms embedded into a framework modules. The performed experiments on real-life and synthetic policies illustrate the relevance and efficiency of our approach for detecting flaws, conflicts and redundancies within acceptable overhead. Moreover, the step-by-step policy analysis depicts the applicability of the semantics-rules to identify in each of the aforementioned cases the contradictions between policies/rules. Please visit http://www.azzammourad.org/#projects to download the framework.

For future work, we can benefit from SBA-XACML to potentially elaborate policy analysis semantics based on the meaning of rules for detecting other types of flaws and identifying access contradictions that can influence the grouping decision between different Web services.

Acknowledgment

This work is supported by the Lebanese American University (LAU) and CNRS, Lebanon.

References

Hussein Jebbaoui received his M.Sc. degree in Computer Science from the Lebanese American University. The topics of his research activities are Web services security and XACML policy evaluation and analysis.

Azzam Mourad is an assistant professor of Computer Science at the Lebanese American University. He holds a Ph.D. in ECE from Concordia University and M.Sc. degree in Computer Science from Laval University. He is currently working on information security, web services, vehicular networks, and formal semantics. He is serving as TPC and reviewers of several prestigious conferences and journals.

Hadi Otrok holds an associate professor position in the Department of ECE at Khalifa University. He received his Ph.D. in ECE from Concordia University. He works on network and computer security, game theory and mechanism design. He chaired several security-related conferences. Moreover, he is a TPC member of several prestigious conferences and reviewer of several IEEE and Elsevier journals.

Ramzi A. Haraty is an associate professor in the Department of Computer Science and Mathematics at the Lebanese American University. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 110 books, book chapters, and journal and conference paper publications.