



Lebanese American University Repository (LAUR)

Conference

Publication metadata

Title: Model-driven specification and design-level analysis of XACML policies

Author(s): Hanine Tout; Azzam Mourad; Chamseddine Talhi

Conference title Second International Conference on Next Generation Computing and Communication Technologies (ICNGCCT 2015)

DOI: <http://dx.doi.org/10.13140/RG.2.1.2573.6167>

Handle: <http://hdl.handle.net/10725/12115>

How to cite this post-print from LAUR:

Tout, H., Mourad, A., & Talhi, C. (2015). Model-driven specification and design-level analysis of XACML policies. In Second International Conference on Next Generation Computing and Communication Technologies (ICNGCCT 2015). DOI, 10.13140/RG.2.1.2573.6167, <http://hdl.handle.net/10725/12115>

© Year 2015

This Open Access post-print is licensed under a Creative Commons Attribution-Non Commercial-No Derivatives (CC-BY-NC-ND 4.0)



This paper is posted at LAU Repository  
For more information, please contact: [archives@lau.edu.lb](mailto:archives@lau.edu.lb)

# Model-Driven Specification and Design-Level Analysis of XACML Policies

Hanine Tout, Azzam Mourad  
Department of Computer Science and Mathematics  
Lebanese American University  
Beirut, Lebanon  
{hanine.tout, azzam.mourad}@lau.edu.lb

Hadi Otrok  
Department of Computer Engineering  
Khalifa University  
Abu Dhabi, UAE  
hadi.otrok@kustar.ac.ae

Chamseddine Talhi  
Department of Software Engineering and IT  
École De Technologie Supérieure  
Montreal, Canada  
chamseddine.talhi@etsmtl.ca

Hamdi Yahyaoui  
Department of Computer Science  
Kuwait University  
Safat, Kuwait  
hamdi@cs.ku.edu.kw

**Abstract**— Throughout the recent years, Web services security has been the target of many researchers. Particularly, by integrating policies and rules to govern the Web services behaviors at runtime, researchers have been able to prove the capability of policy languages in enforcing Web services security. XACML or eXtensible Access Control Markup Language is one of the most widely adopted security standards for controlling access to individual and between composed services based on policies specifications. However, like any other policy language, XACML policies are specified in structural files with complex syntax, which makes the policies specification process both, time consuming and error prone. Moreover, security policies are commonly verified in an afterthought stage after their enforcement, yet with diversity of rules and conditions specified in the policies, hidden conflicts, redundancies and access flaws are more likely to arise, which expose the system to serious vulnerabilities at execution time. To address these problems, we propose in this paper a novel approach that allows high-level specification of XACML security policies and provides design-level analysis to detect problems and vulnerabilities in the policies semantics, a priori to their integration and execution in the system.

**Index Terms**—Web Services Security, XACML, Security Policies, Model-Driven Specification, Design-Level Analysis.

## I. INTRODUCTION

Managing Web services security by policies enforcement has become one of the most active research areas. XACML [1] or eXtensible Access Control Markup Language is one of the most widely adopted security standards for controlling access to individual and between composed services based on policies specifications. However, like any other policy language, XACML policies are specified in structured files of too low and complex syntax, which makes them hard to be used by wide spectrum of users who are accustomed to work with abstract architectural system models. In addition, this also makes the policies definition process time consuming, and

foremost error-prone, especially when combining many policies, rules and conditions to govern the system. In this context, several researchers [2, 3, 4] have proposed UML profiles to offer high-level graphical modeling approach for policies specification. UML profile [5] is one of the UML extension mechanisms that allows UML models to be customized for specific domains, and these approaches have proved its capabilities to define different access control models. Yet, the proposed profiles in these approaches [3, 4] do not cover all the elements of XACML policies and most importantly, they rely on XACML 2.0, which is not the latest version of XACML.

Moreover, with diversity of rules and conditions specified in complex policies, hidden conflicts, redundancies and access flaws are more likely to arise. A conflict between two policy rules arises when they are defined in a way that one of them grants access and the other denies access for the same set of subject(s), resource(s) and action(s). Whereas, two policy rules are redundant when they are defined for the same set of subject(s), resource(s), action(s) and the same set of conditions, with same effect (i.e., deny or permit). Finally, access flaws are badly defined rules and/or policies which allow users to gain accidental access to particular resources. An afterthought analysis of policies after their integration, increases the possibility to propagate these issues through the system deployment where locating and resolving them will be impossible. In this context, different approaches [6, 7, 8] have proposed analysis mechanisms for XACML policies. Yet, these approaches miss important elements in XACML, disregard some of these serious issues, and more importantly none of the proposed analysis mechanisms is applied at the design level, where only the evaluation of policies has been presented [9, 10].

To address all these problems, we present in this paper a novel approach that consists of UML profile to allow high-level, straightforward, visualized specification of standard

XACML policies conforming with the latest language version, and design-level analysis to detect problems like conflicts and redundancies and other vulnerabilities as access flaws in the policies semantics, at design level, a priori to their integration and execution in the system.

The main contributions of this work are twofold:

- UML profile for the latest version of XACML to provide high-level specification of security policies.
- Design-level analysis to detect problems and vulnerabilities like conflicts, redundancies, and access flaws in the policies semantics.

The rest of the paper is organized as follows. Section II presents an overview about the proposed approach architecture, illustrates the proposed UML profile and introduces the design level analysis of policies semantics. Section III demonstrates the feasibility and efficiency of our proposition through a case study. In Section IV, we discuss existing relevant works in the literature to distinguish and shed the light on our contributions. Finally, in Section V, we conclude the paper and draw some future research directions.

## II. PROPOSED APPROACH

The proposed approach architecture is depicted in Fig. 1. We introduce first a UML profile for high-level policies specification of policies, as an alternative to the XML-structured files of XACML. The proposed profile captures all the elements of the latest version of this language (i.e., Policy set, policy, rule, target, combining algorithms, condition, obligation and advice). To specify security policies, users create a UML model and then apply the proposed profile on it by attaching stereotypes, parameterized by tagged values, to the UML elements in the model (M). Using our XACML model to sets converter, the corresponding sets are generated then conveyed to the analyzer. In the latter, we implement algorithms capable of detecting problems and vulnerabilities like conflicts, redundancies and access flaws in the policies semantics at the model level, preventing the integration of such problems in the system at runtime. The analyzer generates a detailed report indicating the problems in case any of them exists, and locating the policies and rules behind them.

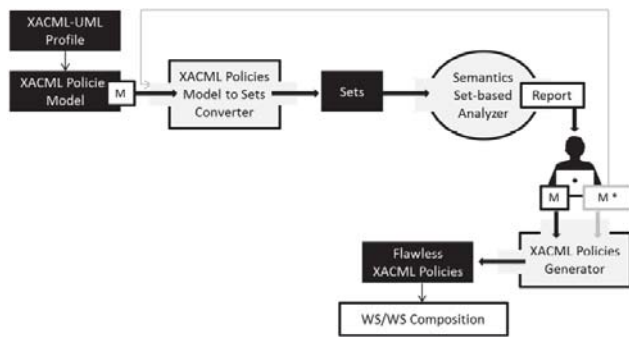


Fig. 1. Approach Architecture

Based on the report, the user updates the model (M\*) for reanalysis. Once the policies model is proved to be flawless, its

corresponding XACML code can be automatically generated using our XACML generator, and finally flawless policies can be applied on to control access whether to individual Web services or even to Web services composition. It's worth mentioning that since the detection is done at the design level, i.e., offline, before the policies enforcement and Web services execution, the proposed analysis do not entail any overhead at runtime.

### A. UML Profile for XACML Policies Specification

In the sequel, we interpret the elements of our proposed profile illustrated in Fig. 2. To remove any ambiguity, we used as much as possible the same names of the elements in the XACML language. We define the appropriate stereotypes, tagged definitions, operations and enumerations to cover all the elements of the latest version of XACML 3.0 that includes new elements and definition capabilities over its predecessor. A *PolicySet*, which extends the Metaclass *Class*, is a container of one or many *Policies*, and has an identifier *ID*, a policy proceeding order *PPO* that determines the order between these policies, and one of the policies combining algorithms *PCA* (i.e., Permit-overrides, Deny-overrides, First-applicable, Only-one-applicable). These algorithms are used in XACML to resolve decision application problems between policies.

Having its own identifier *ID*, a *Policy*, may include many Rules with precedence order among them *RPO* and one of the rule combining algorithms *RCA* (i.e., Permit-overrides, Deny-overrides, First-applicable) to resolve decision problems between its rules. Each rule can have a *Condition*, which is a function that should be validated before applying the rule.

*PolicySet*, *Policy*, and *Rule* can be all associated with *Targets*, *Obligations* and *Advices*. A *Target* identifies the action that a subject can exercise on certain resource, where in our case the action is an invoke and the resource is a service offered by partner Web service. The *Obligation* is an action to be taken *Operation(params)* when certain trigger condition *TriggerCond* is met, which is the rule effect (i.e., Permit or Deny). Finally, the *Advice* an analogous *Obligation*, yet its common use is to explain why someone was denied access to certain resource.

### B. Design-Level Analysis of XACML Policies

Before starting the analysis, the XACML model to sets generator takes the policies UML model defined by the user after applying the proposed profile, and then parses the elements in the model and generates the appropriate sets. These sets form the input to the analysis algorithms capable of detecting conflicts, access flaws and redundancies in the design model. They are defined as follows:

$$PS = \{ID, SOP, PPO, PCA, OBLs, ADs, TAR\}$$

The first generated set is the policy set *PS*, it includes its identifier *ID*, references to the set of policies it contains *SOP*, the order between the policies *PPO*, the combining algorithm *PCA*, sets of obligations *OBLs* and Advices *ADs* if any, and finally the target *TAR* defined as another set of subject *S*, resource *RES* and action *A*.

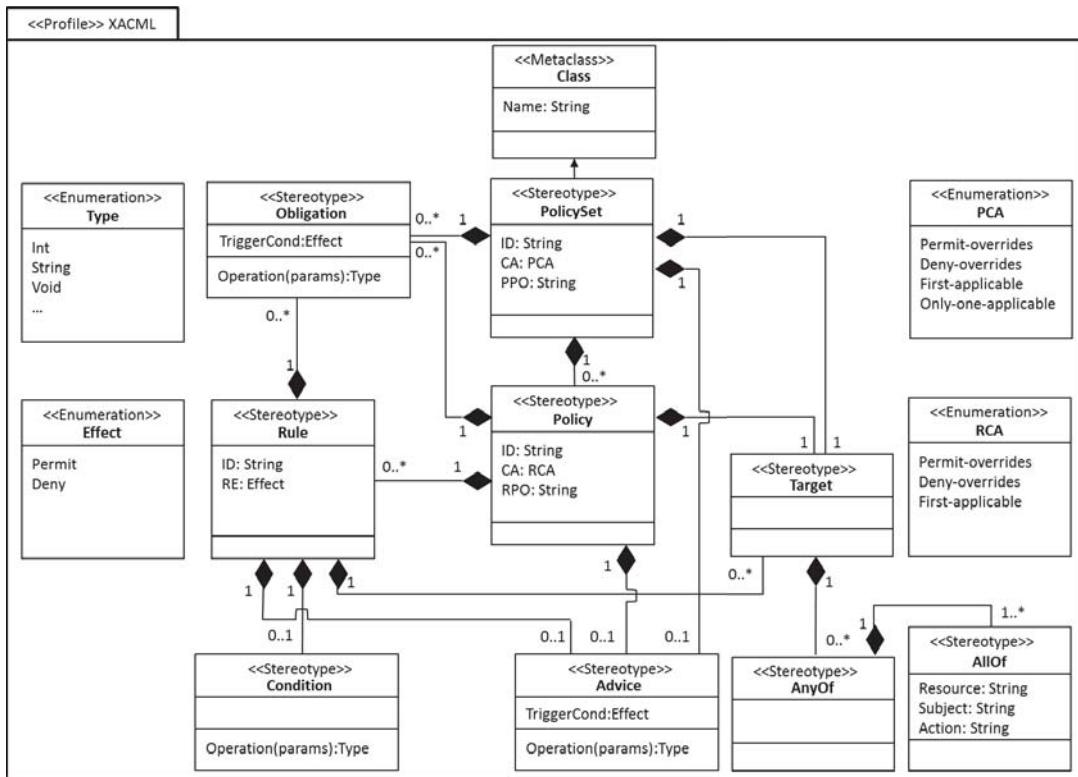


Fig. 2. XACML-UML Profile

$$TAR = \{S, RES, A\}$$

Next, the policy set is generated. Other than the policy *ID*, this set includes references to the set of corresponding rules *SOR*, precedence order between them *RPO*, combining algorithm *RCA*, sets of obligations *OBLs* and Advices *ADs* if any, and the target *TAR*.

$$P = \{ID, SOR, RPO, RCA, OBLs, ADs, TAR\}$$

Finally comes the rule set *R*, which includes *ID*, condition *C*, sets of obligations *OBLs* and Advices *ADs* if any, target *TAR*, and rule effect *E*.

$$R = \{ID, C, OBLs, ADs, TAR, E\}$$

*C* is a function to be evaluated against the target elements (i.e., subject, resource and action), which is defined as

$$C = \{Operation, \{params\}\}$$

Both *OBLs* and *ADs* are defined in the same way.

The analysis is done at three levels; rule-based analysis in Algorithm 1, policy-based analysis in Algorithm 2 and policy set-based analysis in Algorithm 3.

Starting with Algorithm 1; to detect existing conflicts, access flaws and redundancies between two rules *R1* and *R2*, the algorithm compares their targets, conditions and effects.

If the target (i.e., Subject, Resource and Action) and condition set of *R1* are subset of those of *R2* (Lines 1 and 2), both rules have the same effect (Line 3), then *R1* is causing access control flaw. If *R1* target intersects with *R2* target (Lines 9 and 10) and both rules have opposite effects (Line 11), then they are conflicting. Otherwise, with same effects the rules are redundant. Finally, if none of the problems is found, an empty set is returned.

**Algorithm 1** Rule\_Based\_Analysis(*R1*, *R2*)

```

Input   : Two Rules R1 with Target TAR1 = {S1,RES1,A1}, condition C1, Effect E1 and R2 with Target TAR2 = {S2,RES2,A2}, condition C2, effect E2
Output  : Rule analysis ∈ {Flaw, Conflict, Redundant or Null}
1: if (S2 ⊆ S1) ∧ (RES2 ⊆ RES1) ∧ (A2 ⊆ A1) then
2:   if (C2 ⊆ C1) then
3:     if (E1 = E2) then
4:       // R2 is a subset of R1
5:       return "Flaw";
6:     end if
7:   end if
8: end if
9: if ((S1 ∩ S2) ≠ ∅) ∧ ((RES1 ∩ RES2) ≠ ∅) ∧ ((A1 ∩ A2) ≠ ∅) then
10:  if ((C1 ∩ C2) ≠ ∅) then
11:   if (E1 ≠ E2) then
12:    return "Conflict";
13:   else
14:    return "Redundant";
15:   end if
16: end if
17: end if
18: return ;

```

**Algorithm 2** Policy\_Based\_Analysis( $P1, P2$ )

```
Input   : Policy P1 with Target TAR1 = {S1,RES1,A1} and P2 with
          Target TAR2 = {S2,RES2,A2}
Output  : Flaw Problem Set FPS, Conflict Problem Set CPS and Redundancy Problem Set RPS

1: //Check rules in each policy
2: for  $l := 1$  to 2 do
3:   for  $i := 1$  to  $P1\_NumberofRules-1$  do
4:     for  $j := 2$  to  $P1\_NumberofRules$  do
5:       if (RULE_BASED_ANALYSIS( $R_{li}, R_{lj}$ ) = "Flaw") then
6:         FPS = FPS  $\cup$  Flaw $_{R_{li}, R_{lj}}$ ;
7:       end if
8:       if (RULE_BASED_ANALYSIS( $R_{li}, R_{lj}$ ) = "Redundant") then
9:         RPS = RPS  $\cup$  Redundant $_{R_{li}, R_{lj}}$ ;
10:      end if
11:      if (RULE_BASED_ANALYSIS( $R_{li}, R_{lj}$ ) = "Conflict") then
12:        CPS = CPS  $\cup$  Conflict $_{R_{li}, R_{lj}}$ ;
13:      end if
14:    end for
15:  end for
16: end for
17: //Check rules in both P1 and P2
18: if ( $RCA_{P1} = RCA_{P2}$ ) then
19:   if ( $(S1 \cap S2) \neq \emptyset \wedge ((RES1 \cap RES2) \neq \emptyset) \wedge ((A1 \cap A2) \neq \emptyset)$ )
then
20:     for  $l := 1$  to  $P1\_NumberofRules$  do
21:       for  $m := 1$  to  $P2\_NumberofRules$  do
22:         if (RULE_BASED_ANALYSIS( $R_l, R_m$ ) = "Flaw") then
23:           FPS = FPS  $\cup$  Flaw $_{R_l, R_m}$ ;
24:           FPS = FPS  $\cup$  Flaw $_{P_1, P_2}$ ;
25:         end if
26:         if (RULE_BASED_ANALYSIS( $R_l, R_m$ ) = "Conflict") then
27:           CPS = CPS  $\cup$  Conflict $_{R_l, R_m}$ ;
28:           CPS = CPS  $\cup$  Conflict $_{P_1, P_2}$ ;
29:         end if
30:         if (RULE_BASED_ANALYSIS( $R_l, R_m$ ) = "Redundant")
then
31:           RPS = RPS  $\cup$  Redundant $_{R_l, R_m}$ ;
32:           RPS = RPS  $\cup$  Redundant $_{P_1, P_2}$ ;
33:         end if
34:       end for
35:     end for
36:   end if
37: end if
38: return ;
```

**Algorithm 3** PolicySet\_Based\_Analysis( $PS$ )

```
Input   : A Policy Set PS
Output  : Analysis report

1: Global FPS =  $\emptyset$ ; RPS =  $\emptyset$ ; CPS =  $\emptyset$ ;
2: for  $i := 1$  to  $PS\_NumberofPolicies-1$  do
3:   for  $j := i + 1$  to  $PS\_NumberofPolicies$  do
4:     PA = POLICY_BASED_ANALYSIS( $P_i, P_j$ );
5:   end for
6: end for
```

Next, Algorithm 2 analyzes the policies. It provides a set of all problems (i.e., access flaws FPS, conflicts CPS and redundancies RPS) that exist between two policies P1 and P2. The first part of the algorithm (Line 1 till 16) checks for flaws, conflicts and redundancies within each policy, while in the second part (Line 17 till 37), this checking is conducted between rules from different policies. The returned values of Algorithm 1 are appended to the appropriate sets in Algorithm 2.

Finally, Algorithm 3 analyzes the policy sets. This algorithm displays all access flaws, conflicts and redundancies between policies and rules existing within a policy set PS. It initializes the corresponding global sets FPS, CPS and RPS (Line 1) and calls Algorithm 2 (Line 2 till 6)

for checking flaws, conflicts and redundancies within each policy and between policies and subsequently append the relevant sets.

## III. CASE STUDY

To better illustrate our approach, we suggest a Flight System (FS) as a running example. The system consists of a composition of three partners Web services. First, a Financial Data WS, which offers access to financial reports. Second, Flight Inquiries WS, which displays the flights with their schedules, available seats and comparable tickets prices, according to the user preferences. Third, a Reservation WS that offers the ability to book a flight ticket.

To enforce security, the system imposes many policies and rules that can be defined in an XACML file. For space restrictions, Fig. 4 depicts only synopsis of such XACML policy set reflecting the complexity in the policies definition. The policy set consists of two policies P1 and P2 and has a permit-overrides combining algorithm. P1 defines two rules R1 and R2. R1 gives only admin the permission to access the financial data while R2 permits anyone to access the same resource. On the other hand, P2 defines two other rules R3 and R4. R3 allows anyone to make reservation in the flight agency system while R4 prevents anyone from making reservation in particular period. Figure 4 shows clearly that specifying security policies in regular XACML XML-based format is subtle and time consuming, and even the analysis of such format is complex. To recall, XACML provides only an evaluation engine at runtime, yet does not have any efficient mechanism to detect problems and vulnerabilities such as conflicts, redundancies and access flaws between policies and rules.

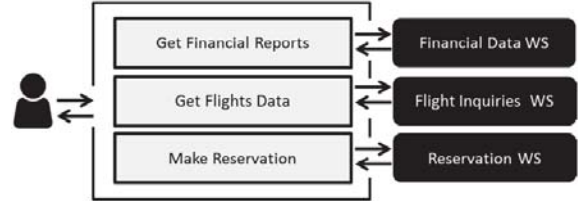


Fig. 3. Flight System (FS)

## A. Model-Driven Security Policies Specification

Rather than writing long, verbose and complex XACML policy set for the Flight System, following our approach, the user can create a simple UML model that contains: Policy set PS1, Policies P1 and P2, Rules R1, R2, R3 and R4, Conditions C1 and C2, and Targets for each rule (along with their sub-elements) then applies systematic transformation on the model based on the proposed UML profile. This is done by:

1. Applying *PolicySet* stereotype on *PS1* and specifying its tagged values *ID*, *CA* and *PPO*.
2. Applying *Policy* stereotype on *P1* and *P2* and specifying their tagged values *ID*, *CA* and *RPO*.
3. Applying *Rule* stereotype on *R1*, *R2*, *R3* and *R4* and specifying their tagged values *ID* and *RE*.



4. Applying *Condition* stereotype on *C1* and *C2* and specifying the appropriate operations *Operation*.
5. Applying *Target* stereotype and its sub-stereotypes on the relevant targets elements and specifying the relevant tagged values of *Subject*, *Resource*, and *Action*.
6. Associate the elements together.

Figures 4a 4b, and 4c depict the model after applying the systematic transformation described above. Due to space restrictions, we split the model on different Figures.

```

<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns="schema:os" PolicyCombiningAlgId="policy-combining-algorithm:permit-overrides" PolicySetId="PS1">
  <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="P1"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides" Version="1.0">
    <Description>financial data policy</Description>
    <Target></Target>
    <Rule Effect="Permit" RuleId="R1">
      <Target> <AnyOf> <AllOf>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            http://localhost:8280/fsservices/getFinancialData/</AttributeValue>
          <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true"></AttributeDesignator>
        </Match> <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">invoke</AttributeValue>
          <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true"></AttributeDesignator>
        </Match> </AllOf> </AnyOf> </Target> <Condition>
          ...
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
          </Apply> ... </Apply> </Condition> </Rule>
        <Rule Effect="Permit" RuleId="R2">
          <Target> ... </Target> </Rule>
      </Policy>
    <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="P2"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides" Version="1.0">
      <Description>reservation policy</Description>
      <Target></Target> <Rule Effect="Permit" RuleId="R3"> <Target> ... </Target> </Rule>
      <Rule Effect="Deny" RuleId="R4"> <Target> ... </Target> <Condition> ...
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
        </Apply> ...
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#time">16:00:00</AttributeValue>
        </Apply> </Apply> </Condition> </Rule>
      </Policy>
    </PolicySet>
  
```

Listing 1. Synopsis of XACML Policy Set for FS

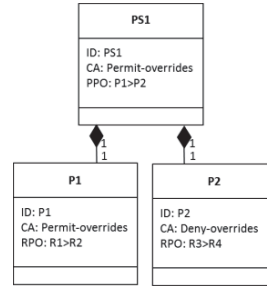
### B. Design-Level Security Policies Analysis

After the specification of security policies and automatic generation of their corresponding sets, the proposed analyzer module takes care of the detection of existing problems and vulnerabilities in the policy set based on the policies semantics. Listing 2 presents a synopsis of the generate analysis report. The highlighted messages illustrate the capability of the proposed algorithms to detect access flaws, conflicts and redundancies between policies and rules in the policy set.

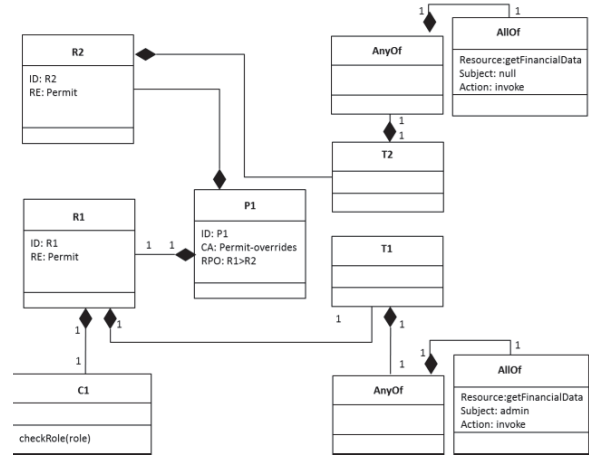
## IV. RELATED WORK

In what follows, we present existing works for model-driven security specification and security policies analysis. Tout et al. [2], the authors proposed a model-driven approach

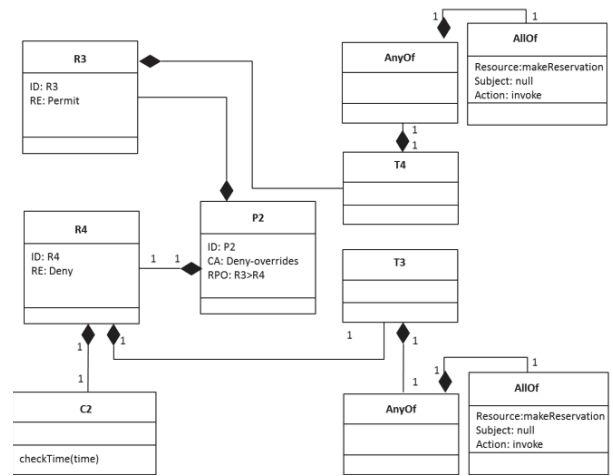
to define and integrate security aspects in Web services composition. They presented a UML profile that extends the BPEL to offer high-level specification of security aspects. Their work relies on specific aspect security language for BPEL, yet our approach relies on the standard XACML language.



(a) Policy Set PS



(b) Policy P1 and its Associated Elements



(c) Policy P2 and its Associated Elements

Fig. 4. Design-Model of Security Policies for FS

```

...
PS1 contains 2 Policies P1 and P2.
Check Access Flaws in P1.
...
R1 and R2 have Equivalent Targets.

```

```

C1 is a Subset of C2.
R1 and R2 have Same Effect, Permit.
Access Flaw Detected between R1 and R2.
FRS = {R1, R2}.
...
Check Conflicts between P1 and P2.
P1 and P2 have Different Targets.
No Conflict between P1 and P2.
Check Conflicts in P2.
P2 contains 2 Rules R3 and R4.
...
R3 and R4 have Equivalent Targets.
C4 is a Subset of C3.
R3 and R4 have Different Effects, Permit and
Deny.
Conflict Detected between R3 and R4.
CRS = {R3, R4}.
...
Check Redundancies in P1.
P1 contains 2 Rules R1 and R2.
...
R1 and R2 have Equivalent Targets.
C1 is a Subset of C2.
R1 and R2 have Same Effect, Permit.
Redundancy Detected between R1 and R2.
RRS = {R1, R2}.

```

Listing 2. Synopsis of the Generated Report

Jin [3] has proposed model-driven architecture to build role based access control (RBAC) model. To address the complexity of XACML XML-based documents, they proposed a UML profile to ease the specification of XACML RBAC applications. Also, Busch et al. [4] argued that XML syntax of XACML makes the process of policies specification difficult and error-prone and thus they proposed a UML-based notation to offer graphical modelling of security properties. However, the notations presented in both approaches did not cover all the elements of XACML policies like obligations, and most importantly, they rely on XACML 2.0, which is not the latest version. Per contra, we presented in this paper a UML profile that covers all the elements of the latest version of XACML, offering the ability to design any policy expressed in this language.

In different works [11, 12], set based approaches have been presented for XACML policies evaluation and analysis, yet not at design-model level. Florian et al. [6] dealt with conflicts, yet did not address other problems like those we presented throughout this paper. Kolovski et al. [7] proposed a formalization of XACML using description logics (DL) and verification using the existing DL verifiers. Even though their analysis is able to detect redundancies between rules, yet they don't provide means for detecting access flaws and even they do not support conditions and some combining algorithms. Rao et al. [8] introduced algebra for fine-grained integration supporting specification of integration constraints. However, they missed many elements of XACML like rule conditions and obligations. Opposed to our work, these approaches cannot support important elements in XACML, discarded some critical problems and vulnerabilities, and more essentially, none of them has proposed analysis at the design level.

## V. CONCLUSION AND FUTURE WORK

This paper presented a UML profile for high-level specification of security policies, and design-level analysis to detect problems and vulnerabilities like conflicts,

redundancies, and access flaws, in the defined policies semantics, a priori to their application at runtime. As for future work, we plan to address different type of flaws that can arise between policies especially those that can threaten more complex systems like Web services composition.

## ACKNOWLEDGMENT

This work has been supported by the Lebanese American University (LAU) and CNRS Lebanon.

## REFERENCES

- [1] Rissanen, Erik. "extensible access control markup language (xacml) version 3.0 (committe specification 01)". Technical report, OASIS, <http://docs.oasisopen.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.pdf>, 2010.
- [2] Tout, Hanine, Azzam Mourad, Hamdi Yahyaoui, Chamseddine Talhi, and Hadi Otrok. "Towards a BPEL model-driven approach for Web services security." In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pp. 120-127. IEEE, 2012.
- [3] Jin, Xin. "Applying model driven architecture approach to model role based access control system." PhD diss., University of Ottawa, 2006.
- [4] Busch, Marianne, Nora Koch, Massimiliano Masi, Rosario Pugliese, and Francesco Tiezzi. "Towards model-driven development of access control policies for web applications." In *Proceedings of the Workshop on Model-Driven Security*, p. 4. ACM, 2012.
- [5] OASIS. *Omg unified modeling language<sup>TM</sup> (omg uml) version 2.5*. Technical report, OASIS, [http://www.sce.carleton.ca/courses/sysc-5708/f14/UML2\\_5-ptc-13-09-05.pdf](http://www.sce.carleton.ca/courses/sysc-5708/f14/UML2_5-ptc-13-09-05.pdf), 2013.
- [6] Huonder, Florian, Josef M. Joller, and Z. H. Rüsçhlikon. "Conflict detection and resolution of XACML policies." *Master's thesis, University of Applied Sciences Rapperswil* (2010).
- [7] Kolovski, Vladimir, James Hendler, and Bijan Parsia. "Analyzing web access control policies." In *Proceedings of the 16th international conference on World Wide Web*, pp. 677-686. ACM, 2007.
- [8] Rao, Prathima, Dan Lin, Elisa Bertino, Ninghui Li, and Jorge Lobo. "An algebra for fine-grained integration of XACML policies." In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pp. 63-72. ACM, 2009.
- [9] Marouf, Said, Mohamed Shehab, Anna Squicciarini, and Smitha Sundareswaran. "Adaptive reordering and clustering-based framework for efficient XACML policy evaluation." *Services Computing, IEEE Transactions on* 4, no. 4 (2011): 300-313.
- [10] Ngo, Canh, Marc X. Makkes, Yuri Demchenko, and Cees de Laat. "Multi-data-types interval decision diagrams for XACML evaluation engine." In *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*, pp. 257-266. IEEE, 2013.
- [11] Mourad, Azzam, and Hussein Jebbaoui. "SBA-XACML: Set-based approach providing efficient policy decision process for accessing Web services." *Expert Systems with Applications* 42, no. 1 (2015): 165-178.
- [12] Jebbaoui, Hussein, Azzam Mourad, Hadi Otrok, and Ramzi Haraty. "Semantics-based approach for detecting flaws, conflicts and redundancies in XACML policies." *Computers & Electrical Engineering* (2015).