Monitoring of Access Control Policy for Refinement and Improvements

Antonello Calabró, Francesca Lonetti, and Eda Marchetti(✉)

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo",
Consiglio Nazionale delle Ricerche (CNR), via G. Moruzzi 1, 56124 Pisa, Italy
{antonello.calabro,francesca.lonetti,eda.marchetti}@isti.cnr.it

Abstract. Access Control is among the most important security mechanisms to put in place in order to secure applications, and XACML is the de facto standard for defining access control policies. As systems and resource utilization evolve, access control policies become increasingly difficult to manage and update according to contextual behaviour. This paper proposes a policy monitoring infrastructure able to identify policy abnormal behaviour and prevent misuse in granting/denying further accesses. This proposal relies on coverage adequacy criteria as well as KPIs definition for assessing the most common usage behaviors and provide feedback for refinement and maintenance of the current access control policy. It integrates a flexible and adaptable event based monitoring facility for run time validation of policy execution. A first validation on an example shows the effectiveness of the proposed approach.

Keywords: Monitoring · Coverage criteria · KPI · Access control policy

1 Introduction

In the Factory of the Future as well as Industry 4.0 more often the adopted context-aware applications rely on a widespread use of wireless sensors and actuators to monitor the process evolution, guide the users interaction and automate the business process at large. The aggregation of high volumes of volatile data and sensitive user information rises a multitude of security and privacy challenges. Typically, large organizations have to rule millions of accesses across thousands of IT resources and sensors. This cause an effective risk of providing users with an excessive or not appropriate access rights over time. In such a situation the proper implementation and enforcement of access control mechanisms becomes imperative. They are able to rule the highly connected and pervasive software intensive systems and prevent unauthorized, erroneous or even malicious usage of critical resources.

Among security mechanisms, a critical role is played by access control systems. These aim at ensuring that only the intended subjects can access the protected data and the subjects get only the permission levels required to accomplish their tasks and no more. To this purpose, access control systems involve

D. Winkler et al. (Eds.): SWQD 2018, LNBIP 302, pp. 17–36, 2018.
https://doi.org/10.1007/978-3-319-71440-0_2

three related activities, namely: *identification* of the user or service requesting an access; *authentication* of the declared identity, usually by means of credentials; and finally *authorization* of authenticated users to perform a set of allowed operations.

Authorization mechanisms are typically based on *access control policies* that rule: the level of confidentiality of data; the procedures for managing data and resources; the classification of resources into category sets yielding different security requirements. Access control policies must be specified and verified with great accuracy: as any error or overlook could result either in forbidding due access rights, or worse in authorizing accesses that should be denied, thus jeopardizing the security of the protected data.

XACML (eXtensible Access Control Markup Language) [1] is a widely used standard language for specifying access control policies. It is an XML-based language conceived with interoperability, extensibility, and distribution in mind, thus enabling the specification of very complex rules. However, such advantages are paid in terms of complexity and verbosity. Writing XACML policies is hard and may be deceiving, as inconsistencies could arise between the security requirements the policy authors intend to specify, and those that the policies actually state. This can easily happen for instance when either some modifications are introduced in a complex policy or when the policy is obtained from the integration of more policies coming from different organizations.

Especially in large scale organizations the common practice, to partially solve this problem, is to define and implement just the basic policies, usually extracted by the internal regulations and requirements that remain unchanged in time. The side effect of such an attitude is that policies could become outdated over time, leading either inconsistencies with the evolved behavioral and technological organization environment, or security vulnerabilities.

So far most, of the research activities have been mainly focused on policy testing [2–5] and only few proposals target the on-line validation and improvement of the policies specifications [6].

The idea of this paper is providing to the policy authors, or more in general to any stakeholder using the policy (generically referred in the following as the policy validators), an integrated dynamic framework to validate and monitor the actual resources access and users behavior. This in order to update, correct or improve the policy specification in granting/denying the accesses.

Through the proposed framework the policy validators may have: (i) a better knowledge of the actual policy usage, (ii) define and on-line calculate possible key performance indicators (KPIs) useful to adjust the policies and (iii) detect possible policy vulnerabilities or outdated rules.

The proposal relies on: (i) the derivation of the relevant coverage information from the policy specification; (ii) the collection of events (requests/responses) during the policy execution by means of a monitoring facility, (iii) and the analysis of them so to collect important policy accesses information and assess defined KPIs.

Indeed monitoring the different requests and responses lets the collection of source data such as: the identification of most accessed resources; the tracking of type of users requiring a resource access; the time and the decision of whether the request is granted or denied and so on. These can be very useful data for modeling the most common user behaviors and detecting possible policy flaws or improvements. Moreover key performance indicators, relative to the criticality of some specific actions or data, can be specified and on-line computed for a better resources access control. For instance possible KPIs could be: the frequency of the resource access by a specific users; the identification of users that most frequently require a specific action; the date and time when the access permission of a specific resource is more frequently activated and so on.

Deviation of the observed data from the boundaries values established for each KPI could indicate abnormal behavior and could required a deeper analysis to identify possibly access control policy modifications or improvements.

The contribution of this paper can be summarized into:

- the integration of a monitoring framework into an access control system architecture;
- the definition of the architecture of the Policy Monitoring Infrastructure enabling: the definition of specific KPIs; the policy parsing so to extract coverage information; the monitoring of requests and response execution so to compute coverage data; the analysis of data collected to assessing their compliance with the access control rules and specified KPI;
- an instantiation of the proposed architecture on the XACML access control language.

An simulation example showing the usage of the proposed monitoring framework is also provided considering a prototyped version of a booking system of the Multimedia Laboratory of Pisa university, used for improving foreign languages or preparing lecture material.

Preliminary results of such example demonstrated the feasibility of the proposal and highlighted some of the potentialities of the Policy Monitor Infrastructure. Indeed without such a feature most of the weaknesses evidenced in the experiment and suggestions for access control policy improvements were not easily identifiable.

The remainder of this paper is structured as follows: Sect. 2 introduces the basic concepts of access control systems and coverage testing; Sect. 3 presents the architecture of the Policy Monitoring Infrastructure; Sect. 4 reports of the usage of the Policy Monitoring Infrastructure on example; Related works are presented in Sect. 5 and finally, Sect. 6 concludes the paper also hinting at discussion and future work.

2 Background

In the following section some basic concepts about XACML access control systems and coverage testing are provided.

2.1 XACML and Access Control Systems

Access control is one of the most adopted security mechanisms for the protection of resources and data against unauthorized, malicious or improper usage or modification. It is based on the implementation of access control policies expressed by a specific standard such for instance the widely adopted eXtensible Access Control Markup Language (XACML) [1].

XACML is a platform-independent XML-based language for the specification of access control policies. Briefly, an XACML policy has a tree structure whose main elements are: PolicySet, Policy, Rule, Target and Condition. The PolicySet includes one or more policies. A Policy contains a Target and one or more rules. The Target specifies a set of constraints on attributes of a given request. The Rule specifies a Target and a Condition containing one or more boolean functions. If the Condition is evaluated to true, then the Rule's Effect (a value of Permit or Deny) is returned, otherwise a NotApplicable decision is formulated (Indeterminate is returned in case of errors). The PolicyCombiningAlgorithm and the RuleCombiningAlgorithm define how to combine the results from multiple policies and rules respectively in order to derive a single access result. The structure of an access control policy and an access control request is sketched in Fig. 1. While an example is provided in Listing 1.

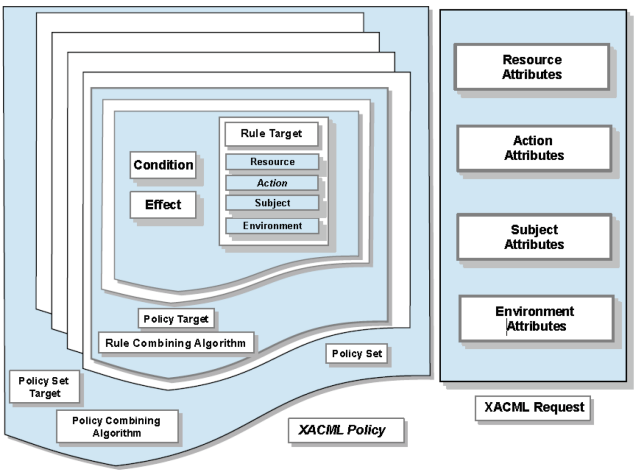

Fig. 1. Anatomy of an XACML policy and an XACML request

The main components of an XACML based access control system are shown in Fig. 2. In particular the Policy Administration Point (PAP) is the system entity in charge of managing the policies; the Policy Enforcement Point (PEP), usually embedded into an application system, receives the access request in its native format, constructs an XACML request and sends it to the Policy Decision Point (PDP); the Policy Information Point (PIP) provides the PDP with

the values of subject, resource, action and environment attributes; the PDP evaluates the policy against the request and returns the response, including the authorization decision to the PEP.

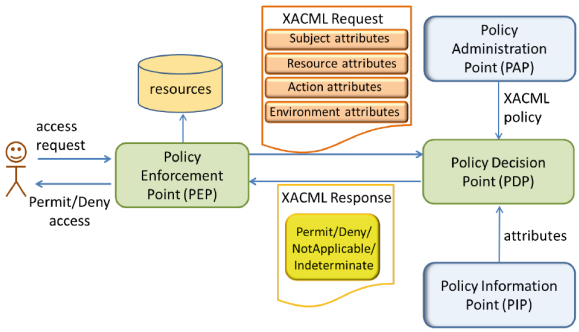

Fig. 2. Access control system architecture

2.2 Adequacy Criteria and Coverage

The notion of adequacy criteria has been extensively investigated in software engineering literature, an in particular for software testing where it is generally used to assess the effectiveness of a test suite [7,8]. In test coverage criteria, a set of requirements that a test suite must fulfill is established and it is mapped onto a set of entities that must be covered when the test cases are executed, as for instance all statements or all branches of a program control-flow. The coverage criterion is satisfied if all the entities are covered; otherwise, the percentage of covered entities represents a quality measure of the test suite.

The intuitive motivation behind measuring test coverage is that if some entity has never been tested, it might contain undetected faults. Obviously, the converse reasoning does not apply: if we had covered all entities and detected no failures, this does not necessarily imply that the program is correct. In a similar way, we propose here to assess the adequacy of the access control policy execution by identifying what are the relevant entities to be covered and by assessing if all of them, or otherwise what percentage, have been executed.

As for test adequacy, the motivation behind assessing access control policy execution adequacy is that if some entities are not covered, we cannot exclude that these might hide some problems, policy incompleteness, entity misuses or security flaw. Similarly to the testing session, i.e. the period along which the test adequacy is measured, the observation window is the observation period associated to the access control policy execution coverage measure.

Intuitively, a sliding observation window over a time measurement unit can be established, which could be either continuous (e.g. the entities covered in the last week) or discrete (e.g., the most recent 15 entities). The proposed access control policy adequacy criterion extends the general monitoring adequacy criterion

presented in [9,10], by defining and implementing an instantiation of this adequacy criterion for the XACML access control policy execution. In particular we consider the following definition:

Definition 1. *Denote $r_i \in R$ the i-th entity to be covered, and by $\delta_i \in \Delta$ the length of its associated observation window. The access control policy execution adequacy criterion C dynamically measures the coverage on R for a given entity i at each time unit t as follows:*

$$C[R, \Delta](t) = \frac{\sum_{i=1}^{|R|} \lambda_i(t)}{|R|}$$

where for $r_i \in R$ and $\delta_i \in \Delta$

$$\lambda_i(t) = \begin{cases} 1 & \text{if } r_i \text{ is covered at least once in } [t - \delta_i, t] \\ 0 & \text{otherwise} \end{cases}.$$

According to this definition the length of δ_i could be different for each r_i, or could be the same for all entities. In summary the definition of access control policy execution adequacy introduces the following concepts:

- an "adequate access control policy execution" is a policy execution on which a set of entities r_i are covered in a window δ_i;
- a Policy Monitoring, i.e. an infrastructure that, at every instant t, can provide a coverage measure as in Definition 1. If this is less than 1, it can provide a list of those entities that have not been covered;
- an entity that is not covered is an entity of the access control policy that has not been executed for some time.

Inside a access control policy execution what is an entity to be covered can be provided at different levels and with different targets [2,11]. In this paper we consider the *XACML smart coverage* approach presented in [11] which focuses on the XACML policy rules coverage. Briefly, the criterion computes the *Rule Target Set*, i.e. the union of the target of the rule, and all enclosing policy and policy sets targets. The main idea is that in order to match the rule target, the requests must first match the enclosing policy and policy sets targets. If the rule target has several subjects, resources, actions, and environments and the enclosing policy and policy set targets are empty, to cover the rule target the request should have specifical structure. It should include a subject contained in target subjects set, a resource contained in the target resources set, an action contained in the target actions set, an environment contained in the target environments set. Finally, if the *Rule Target Set* of a rule is empty and its condition is evaluated to True or False, all requests are covering this rule. We refer to [11] for further details.

In this paper, according to this criterion, we adopted the following definitions:

Definition 2 (Rule Entity). *Given a XACML access control policy, a Rule R_i and its* Rule Target Set*, a rule entity RE_i is the couple (*Rule Target Set*,* Rule Verdict*) where the* Rule Verdict *is the verdict associated to the Rule R_i when its condition is evaluated to True.*

Definition 3 (Rule Coverage Domain). *Considering a XACML access control policy, the Rule Coverage Domain is the set of all the Rule Entities of the policy.*

Definition 4 (Percentage of Rule Coverage). *With reference to Definition 1, the percentage of rule coverage at time t is given by 100*C, where R is the Rule coverage domain.*

Consequently, at a given instant a XACML access control policy execution is adequated with respect to the rule coverage criterion if the percentage of Rule Entities covered is 100% (or greater than an established threshold level).

3 Policy Monitoring Infrastructure

With reference to Fig. 3, we present in this section the components of the Policy Monitoring Infrastructure. In particular:

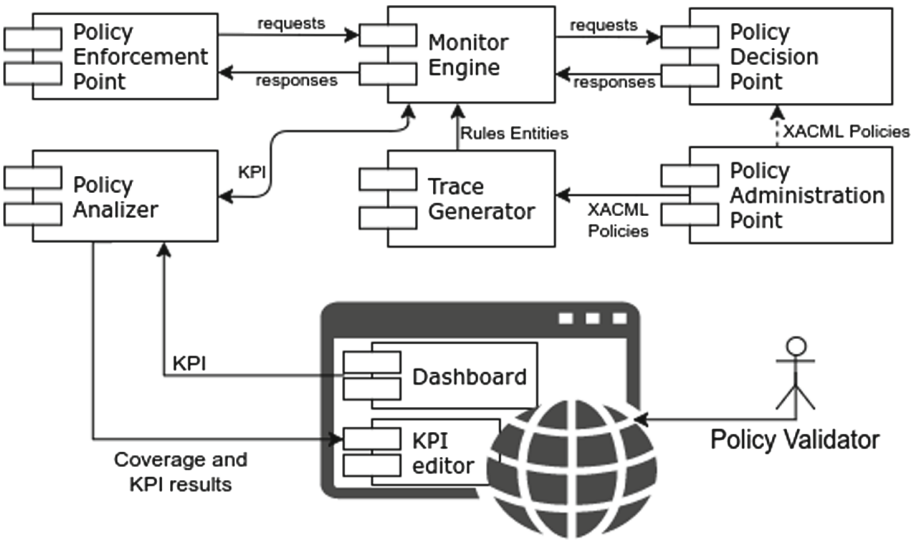

Fig. 3. Policy monitoring infrastructure

- *Policy Enforcement Point (PEP).* It is usually embedded into an application system, receives the access request in its native format, constructs an XACML request and sends it to the Policy Decision Point (PDP) through the *Monitor Engine*;
- *Policy Administration Point (PAP).* It is the system entity in charge of storing and managing the XACML policies. It sends the policy both to the Policy Decision Point (PDP) for its evaluation and to the *Trace Generator* for the traces extraction.

- *Policy Decision Point (PDP).* It evaluates the policy against the request and returns the response, including the authorization decision, to the *Monitoring Engine*. It communicates with the *Monitoring Engine* though a dedicated interface such as a REST one.
- *Trace generator.* It is in charge of implementing the proposed access control policy adequacy criterion. It takes in input the policy from the *Policy Administration Point* and, according to the coverage criterion, derives all the possible Rule Entities. Usually, the Rule Entities extraction is realized by an optimized unfolding algorithm that exploits the policy language structure. Intuitively, the main goal is to derive an acyclic graph, defining a partial order on policy elements. Several proposals are available such as [2,11] for XACML policy specification. Once extracted, the Rule Entities are provided to the *Monitor Engine*.
- *Monitor Engine.* It is in charge of collecting data of interest during the runtime policy execution. There can be different solutions for monitoring activity. In this paper we rely on Glimpse [12] infrastructure which has the peculiarity of decoupling the events specification from their collection and processing. The main components of the *Monitoring Engine* are: (i) the *Complex Events Processor (CEP)* which analyzes the events and correlates them to infer more complex events; (ii) the *Rules Generator* that generates the rules using the rule templates starting from the derived Rule Entities to be monitored. A generic rule consists of two main parts: the events to be matched and the constraints to be verified, and the events/actions to be notified after the rule evaluation; We refer to [12] for a more detailed description of the Glimpse architecture.
- *Policy Analyzer.* It is in charge of the final derivation of the KPIs values according to the covered and non covered Rule Entities. Moreover, according to the coverage criterion it is able to identify the Rule Entities of the policy that could generate flaws or security violations, providing hints to the Policy Validator for policy improvement or refinement.
- *GUI.* It allows to define the KPIs to be monitored (*KPI editor*) and to visualize to the Policy Validators the collected coverage data and KPI values (*Dashboard*) so to let them to analyze and possibly refine the policy. KPI Editor exploits the Domain Specific Language feature of the Drools technologies. In particuar it extracts from the policy specificion some set of values (for instance enviroment condition parameters, subjects, resources, actions an other target contraints) and provides a set of predefined fuctions and logical operators useful for the definition of the KPIs. The purpose is to let the user able to compose rules using a familiar set of logical expressions without requiring knowledge of the implementation language of the Monitor Engine.

4 Example

In this section we present an example of instantiation and use of the Policy Monitoring Infrastructure depicted in Sect. 3. In particular, in the following

subsections we briefly introduce the example considered (Sect. 4.1) and discuss the results collected (Sect. 4.2).

4.1 Example Description

The example considered in this paper is a prototyped version of a booking system of the Multimedia Laboratory of Pisa university, used for improving foreign languages or preparing lecture material. Since the number of stations of this laboratory is limited to 25, the access to different types of users (Master Students, PhD Students, Professors) during the working day is regulated by an access control policy which automatically distributes the booking requests over three time slots. Listing 1 presents the simplified version of the XACML access control policy used for booking a station in the Multimedia Laboratory. Specifically, the policy includes a policy target (lines 8–29) allowing the access only for booking the *MultimediaLab* resource. A first rule (ruleA) (lines 30–64) specifies that Master Students can book the lab starting from 9 am for 4 h. A second rule (ruleB) (lines 65–99) specifies that PHD Students can book the lab starting from 14 pm for 4 h, while a third rule (ruleC) (lines 100–134) specifies that Professors can book the lab only after 18 pm. Finally, the default rule (line 135) denies the access in the other cases. Users of the Multimedia Laboratory are not aware of this internal access control policy and through associated PEP they can ask for booking in all the working day time slots.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
3   xmlns:xacmlcontext="urn:oasis:names:tc:xacml:2.0:context:schema:os"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os␣http://
        docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
6   PolicyId="MultimediaLabPolicy"
7   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
        algorithm:first-applicable">
8    <Target>
9     <Resources>
10     <Resource>
11      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
            equal">
12       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             MultimediaLab</AttributeValue>
13       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0
             :resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#
             string"/>
14      </ResourceMatch>
15     </Resource>
16    </Resources>
17    <Actions>
18     <Action>
19      <ActionMatch
20       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
21       <AttributeValue
22        DataType="http://www.w3.org/2001/XMLSchema#string">booking</
              AttributeValue>
23       <ActionAttributeDesignator
24        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
25        DataType="http://www.w3.org/2001/XMLSchema#string"/>
26      </ActionMatch>
27     </Action>
28    </Actions>
```

```
  </Target>
  <Rule RuleId="rule1" Effect="Permit">
   <Description> A subject who is a Master Student can book the MultimediaLab
        in the morning.</Description>
   <Target>
    <Subjects>
     <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
          equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >MasterStudent</AttributeValue>
       <SubjectAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
     </Subject>
    </Subjects>
    <Environments>
     <Environment>
      <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:time-
          equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09
           :00:00</AttributeValue>
       <EnvironmentAttributeDesignator
        AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
        DataType="http://www.w3.org/2001/XMLSchema#time"/>
      </EnvironmentMatch>
     </Environment>
     <Environment>
      <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0
          :function:dayTimeDuration-equal">
       <AttributeValue
        DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
            dayTimeDuration">PT4H</AttributeValue>
       <EnvironmentAttributeDesignator
        AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
        DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
            dayTimeDuration"/>
      </EnvironmentMatch>
     </Environment>
    </Environments>
   </Target>
  </Rule>
  <Rule RuleId="ruleB" Effect="Permit">
  <Description> A subject who is a PhD Student can book the MultimediaLab in
      the afternoon.</Description>
  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
         equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
       >PHDStudent</AttributeValue>
      <SubjectAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
    </Subject>
   </Subjects>
   <Environments>
    <Environment>
     <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:time-
         equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">14
          :00:00</AttributeValue>
      <EnvironmentAttributeDesignator
       AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
       DataType="http://www.w3.org/2001/XMLSchema#time"/>
```

```
86        </EnvironmentMatch>
87       </Environment>
88       <Environment>
89        <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0
              :function:dayTimeDuration-equal">
90         <AttributeValue
91          DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
                dayTimeDuration">PT4H</AttributeValue>
92         <EnvironmentAttributeDesignator
93          AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
94          DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
                dayTimeDuration"/>
95        </EnvironmentMatch>
96       </Environment>
97      </Environments>
98     </Target>
99    </Rule>
100   <Rule RuleId="ruleC" Effect="Permit">
101   <Description> A subject who is a Professor can book the MultimediaLab in
          the evening.</Description>
102   <Target>
103    <Subjects>
104     <Subject>
105      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
             equal">
106       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
107        >Professor</AttributeValue>
108       <SubjectAttributeDesignator
109        AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
110        DataType="http://www.w3.org/2001/XMLSchema#string"/>
111      </SubjectMatch>
112     </Subject>
113    </Subjects>
114    <Environments>
115     <Environment>
116      <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:time-
             equal">
117       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">18
              :00:00</AttributeValue>
118       <EnvironmentAttributeDesignator
119        AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
120        DataType="http://www.w3.org/2001/XMLSchema#time"/>
121      </EnvironmentMatch>
122     </Environment>
123     <Environment>
124      <EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0
             :function:dayTimeDuration-equal">
125       <AttributeValue
126        DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
                dayTimeDuration">PT4H</AttributeValue>
127       <EnvironmentAttributeDesignator
128        AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-time"
129        DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#
                dayTimeDuration"/>
130      </EnvironmentMatch>
131     </Environment>
132    </Environments>
133   </Target>
134   </Rule>
135   <Rule RuleId="default" Effect="Deny"/>
136 </Policy>
```

Listing 1. MultimediaLab access policy

In this section we show the use of the Policy Monitoring Infrastructure schematize in Fig. 3, for analyzing the load of the booking Multimedia Laboratory system during a working day. For this we set up an experiment in which we simulated the behavior of different types of users (Master Students, PhD Students, Professors) in booking the Multimedia Laboratory according to their preferred time slots. In particular considering the Definition 2 of Sect. 2 and the policy of Listing 1, through

the *Trace Generator* component, the Policy Monitoring Infrastructure automatically derives the set of Rule Entities as reported in Table 1.

Successively, thought the KPI Editor of the Policy Monitoring Infrastructure GUI, we defined the following KPIs:

- KPI1 = daily percentage the overall accesses of Master Students, PhD Students and Professors greater than 60%
- KPI1.1 = daily percentage of accesses of Master Students greater than 36%
- KPI1.2 = daily percentage of accesses of PhD Students greater than 6%
- KPI1.3 = daily percentage of accesses of Professors greater than 18%
- KPI2 = daily number of allowed requests in each time slot less or equal than 25.

From an implementation point of view the above defined KPIs have been translated into the following rules:

- KPI1 = daily percentage of coverage of RE_1 RE_2 and RE_3 greater than 60%
- KPI1.1 = daily percentage of coverage of RE_1 greater than 36%
- KPI1.2 = daily percentage of coverage of RE_2 greater than 6%
- KPI1.3 = daily percentage of coverage of RE_3 greater than 18%
- KPI2 = daily number of allowed requests in each time slot less or equal than 25.

Table 1. *Rule Coverage Domain* of Listing 1

$RE_1 = \{(\emptyset, \{MultimediaLab\}, \{booking\}, \emptyset), (\{MasterStudent\}, \emptyset, \emptyset, 9-13), Permit\}$
$RE_2 = \{(\emptyset, \{MultimediaLab\}, \{booking\}, \emptyset), (\{PhDStudent\}, \emptyset, \emptyset, 13-17), Permit\}$
$RE_3 = \{(\emptyset, \{MultimediaLab\}, \{booking\}, \emptyset), (\{Professor\}, \emptyset, \emptyset, 17-21), Permit\}$
$RE_4 = \{(\emptyset, \{MultimediaLab\}, \{booking\}, \emptyset), (\emptyset, \emptyset, \emptyset, \emptyset), Deny\}$

Finally to set up the simulation environment we established the frequency of a booking request of the three type of users. For this we took the data relative to the requests to the Multimedia Laboratory booking system of the last five years from people belonging to the Software Engineering course of University of Pisa. Then we derived the following percentages: 60%, 30%, 10% for Master Students, PhD Students, Professors respectively. Moreover, we made interviews to a sample of Professors (10), Master Students (60) and PhD Students (30) asking them their preferred time slots. According to their preferences, we derived the average values of frequencies of preferred time slots as shown in Table 2.

The collected information has been used to derive the sample of booking requests useful for the simulation experiment as reported in the last column of Table 2. Specifically for each of the time slot and according to the computed percentages distribution, we manually derived an overall sample of 1000 XACML requests distributed as in the Table. Each of these requests includes a subject

(values chosen from Master Student, PhD Student, Professor), a resource (MultimediaLab), an action (booking) and a time slot (values chosen from 17–21, 13–17, 9–13).

Knowing that the booking system of the Multimedia Laboratory receives on average 50 booking requests per day, through the Policy Monitoring Infrastructure GUI, we set the observation windows of our simulation experiment to a day long (see Definition 1 in Sect. 2). Then we randomly selected, from the generated 1000 XACML requests, sets of 50 requests. We repeated the selection considering a period of 5 weeks, i.e. 25 observation windows, for a overall number of 1250 executed requests.

Bypassing in this simulation experiment the PEP Component, the Monitor Engine Component sent the requests of one by one to the PDP of the Multimedia Laboratory booking system and collected the coverage data.

Table 2. Request frequency

Type of users	Percentage frequency	Time slot	Preferred percentage	# Generated requests
Master students	60%	17–21	60%	360
		13–17	35%	210
		9–13	5%	30
Professors	10%	17–21	5%	5
		13–17	80%	80
		9–13	15%	15
PhD students	30%	17–21	20%	60
		13–17	70%	210
		9–13	10%	30

4.2 Results

In this section we report the results obtained from the simulation experiment. As shown in Table 3, we run on the PDP 50 randomly selected requests per day and we collected, through the Monitor Engine, the data useful for evaluating the KPIs and deriving interesting information on the Multimedia Laboratory booking profiles.

Specifically in the first column there is the incremental number of observation windows considered in the experiment (25 days). From the second to the forth column (from the fifth to the seventh and from the eighth to the tenth) there are the data relative to the Master Students (Professors and PhD Students respectively) booking requests for defined time slots (17–21, 13–17, 9–13). These data have been collected by the Monitor Engine component by comparing and matching the Rule Entities values with the XACML requests and the corresponding PDP responses. These values have been then refined and analyzed the Policy Analyzer Component to derive the KPIs values as reported in Table 3.

In particular in the last four columns there are the daily values computed for the KPI1, KPI1.1, KPI1.2 and KPI1.3 respectively. Finally in the columns highlighted in gray (fourth, fifth, ninth column with bottom label KPI2) there are the comupted KPI2 values corresponding to the daily number of allowed requests in each time slot. Average percentage value for each column is provided in the last row. Results of Table 3 are provided to the Policy Validator through the Dashboard component of the Policy Monitoring Infrastructure GUI.

From the analysis of the data collected during the monitoring activity the Policy Validator could derive differen information such for instance:

- the KPI1 is never satisfied because in each observation windows the coverage percentage is always less than 41% (maximum reached value) with an average value of 29.07% quite far from the established boundary of 60%. This means that the current implemented access control policy does not mach the real booking behavior of the three different users: few booking requests accepted and an extremely high value rejected. Indeed (by difference) the average percentage of coverage of Rule Entity RE_4 is 70.93%. This situation evidences a pressing need to improve the access control policy to better satisfy the real users behavior.
- From a detailed analysis of the data collected of KPI1.1, KPI1.2 and KPI1.3 emerges that the failure of KPI1 is mainly due to the behavior of Master Students and Professors. Indeed, while KPI1.3 is most of the times satisfied (a part from Day 9 and Day 19 where the percentage of coverage are 16% and 13.33% respectively) demonstrating a good ruling of booking access for the PhD Students, KPI1.1, KPI1.2 are never verified. Percentage of accepted booking requests for Master Students rarely reaches values greater than 6% (only Days 10 and 21) and is on average 2.55%. These values are very far from the established 36% of the KPI1.1. Almost the same situation can be experienced for Professors, where the greater value for KPI1.2 is 4.17% with an average of 0.82%. Therefore, improvements on the access control policy for the booking slot times for Master Students and Professors are necessary.
- Considering the daily number of allowed requests in each time slot (i.e. fourth, fifth, ninth column with bottom label KPI2) the values are always less or equal than the established 25 and therefore KPI2 is always satisfied. However, the number of allowed requests for PhD Students are always between 8 and 19, with an average of 12, meaning a quite good resource allocation. While, for Master Students and Professors the situation is quite different. Allowed booking requests for Master Students varies from 0 to 4 with an average of 1.2; for Professors varies from 0 to 2 with an average of 0.4. This means that in the morning and evening time slot the Multimedia Laboratory is almost empty. This is in line with the results obtained for KPI1.1, KPI1.2 and KPI1.3.

The KPIs analysis stressed the exigence of an access control policy improvement specifically for the Master Students and Professors. Suggestions come from a detailed analysis of the requests covering the Rule Entity RE_4, i.e. the requests denied by the PDP. In collecting coverage and KPIs data, the Policy Monitoring Infrastructure provides to the Policy Validator additional information about the

nature of the denied requests. Specifically the Policy Monitoring Infrastructure evidences which kind of subject is booking the Multimedia Laboratory and at what time. These data are reported in Table 3 in the second and third column for Master Students, in the sixth and seventh column for Professors, and in the eighth and tenth column for PhD Students. Analyzing this information the Policy Validator can observe that most of the requests from Master Students are associated to the time slot 17–21, while few variations in booking activity can be observed for the Professors (just very small increase for the 9–13 time slot).

An immediate possible improvement is therefore to swap the time slots implemented in the access control policy for Master Students and Professors i.e. changing in the first rule (ruleA) at lines 47 from 9 to 17, and in the third rule (ruleC) at line 117 from 17 to 9.

Table 3. Experiment results: first round

	Master Student			Professor			PhD Student						
Day	17-21	13-17	9-13	17-21	13-17	9-13	17-21	13-17	9-13	% KPI1	% KPI1.1	% KPI1.2	% KPI1.3
1	14	12	2	0	0	0	2	14	2	34.78	4.35	0	30.43
2	17	10	1	2	2	0	2	12	2	31.25	2.08	4.17	25
3	10	14	3	0	0	0	4	14	0	37.78	6.67	0	31.11
4	9	11	0	0	0	2	3	18	2	40	0	0	40
5	17	10	0	2	2	0	1	15	2	34.7	0	4.08	30.61
6	18	5	0	0	0	0	6	15	2	32.61	0	0	32.61
7	21	6	1	0	0	0	1	19	1	40.82	2.04	0	38.78
8	13	12	0	0	0	1	2	11	4	25.58	0	0	25.58
9	23	10	1	2	2	0	4	8	0	22	2	4	16
10	20	9	3	0	0	2	2	9	1	26.09	6.52	0	19.57
11	17	6	2	2	2	0	5	15	0	38.78	4.08	4.08	30.61
12	16	11	0	1	1	1	4	11	1	26.09	0	2.17	23.91
13	14	8	2	0	0	3	4	15	1	36.17	4.26	0	31.91
14	23	8	1	0	0	1	2	11	1	25.53	2.13	0	23.40
15	19	10	1	0	0	1	3	9	2	22.22	2.22	0	20
16	14	14	1	1	1	2	3	10	2	25	2.08	2.08	20.83
17	22	10	1	0	0	0	5	10	0	22.92	2.08	0	20.83
18	17	12	1	0	0	0	2	12	2	28.26	2.17	0	26.09
19	18	14	0	0	0	1	4	6	2	13.33	0	0	13.33
20	22	9	1	0	0	2	3	9	1	21.28	2.13	0	19.15
21	23	8	4	0	0	0	5	9	1	26	8	0	18
22	12	17	1	0	0	2	1	8	2	20.93	2.33	0	18.60
23	17	11	2	0	0	1	1	13	1	32.61	4.35	0	28.26
24	15	13	0	0	0	0	0	18	0	39.13	0	0	39.13
25	14	19	2	0	0	0	2	9	2	22.92	4.17	0	18.75
average	17	10.76	1.2	0.4	0.4	0.76	2.84	12	1.36	29.07	2.55	0.82	25.70
			KPI2	KPI2				KPI2					

For aim of completeness we performed a second simulation experiment using the same setting of the previous one but with the access control policy modified as described above. In Table 3 we reported the obtained results.

From the analysis of the new results the Policy Validator could observe that:

– Excluding 6 observation windows, where the coverage percentage is in any case greater than 54%, the KPI1 is almost satisfied with an average value of

Table 4. Experiment results: second round

	Master Student			Professor			PhD Student						
Day	17-21	13-17	9-13	17-21	13-17	9-13	17-21	13-17	9-13	% KPI1	% KPI1.1	% KPI1.2	% KPI1.3
1	21	9	3	0	0	0	2	12	1	68.75	43.75	0	25
2	20	14	1	0	0	0	3	8	0	60.87	43.48	0	17.39
3	20	9	0	0	0	0	2	14	4	69.39	40.82	0	28.57
4	16	13	1	0	0	0	4	10	3	55.32	34.04	0	21.28
5	17	11	1	0	0	0	2	9	2	61.91	40.48	0	21.43
6	16	11	1	0	0	2	2	13	2	65.96	34.04	4.26	27.66
7	16	13	4	0	0	0	0	11	1	60	35.56	0	24.44
8	15	8	3	0	0	2	4	12	1	64.44	33.33	4.44	26.67
9	18	12	3	1	1	0	1	9	2	57.45	38.30	0	19.15
10	15	9	0	0	0	2	2	17	1	73.91	32.61	4.35	36.96
11	19	11	0	0	0	2	1	14	1	72.92	39.58	4.17	29.17
12	24	9	2	0	0	1	3	6	2	65.96	51.06	12.13	12.77
13	16	11	0	0	0	0	3	12	1	65.12	37.21	0	27.91
14	12	8	2	1	1	0	7	14	1	56.52	26.09	0	30.44
15	21	7	0	1	1	0	2	16	1	75.51	42.86	0	32.65
16	17	13	2	0	0	1	1	15	0	67.35	34.69	20.04	30.61
18	20	10	1	0	0	1	2	11	2	68.10	42.55	2.13	23.40
19	18	10	2	0	0	3	5	5	1	59.09	40.91	6.82	11.36
20	18	10	2	0	0	2	1	12	1	69.57	39.13	4.35	26.09
21	18	14	1	0	0	1	5	7	2	54.17	37.05	2.08	14.58
22	19	9	1	0	0	0	5	12	2	64.58	39.58	0	25
23	18	9	2	0	0	0	0	11	3	67.44	41.86	0	25.58
24	20	8	2	1	1	0	4	12	0	66.67	41.67	0	25
25	17	13	2	0	0	0	5	7	0	54.55	38.64	0	15.91
average	18.12	10.32	1.44	0.16	0.16	0.76	2.72	11.08	1.44	64.80	39.24	1.66	23.91
	KPI2					KPI2		KPI2					

64.80%. The implemented changes in the access control policy provide a big improvement to this KPI and confirm that now the access control policy is more close to the real booking behavior of different users (Table 4).

– From the detailed analysis of the data collected for KPI1.1, KPI1.2 and KPI1.3 emerges that: KPI1.3 is still most of the times satisfied (a part from 5 observation windows where the percentage of coverage in any case greater than 11%); percentage of accepted booking requests from Master Students is increased on average (average value is 39.24%) and only in 7 observation windows the value is less than the boundary 36%, but in any case greater than 26%. The KPI1.2 is still not satisfied with values low than the established 6%. In such specific case, by analyzing the data of Professors booking requests (see fiftht and seventh column) it is evident that in any case Professors rarely ask for booking Multimedia Laboratory. To improve this situation alternative way of modifying the access policy should be considered.
– The daily number of allowed requests in each time slot (i.e. second, seventh and ninth column with bottom label KPI2) are still always less or equal than 25 and therefore KPI2 always satisfied. However, the number of allowed requests for Master Students is improved a lot with values varying from 12 to 24 with an average of 18,12. This means a good improvement for the resource

allocation for Master and PhD Students even if the situation for Professors remain almost unchanged.

Of course further improvements to the access control policy could be possible in particular for improving the resource allocation during the 9-13 slot where the Multimedia Laboratory remains almost empty. For instance providing parallel booking slot sessions and so on. However this would be a simple experiment to illustrate the importance and the potentialities of a Policy Monitoring Infrastructure inside an access control systems. Indeed without such a feature the analyzed data were not available and improvements not easily identifiable.

5 Related Works

This work spans over several research directions, including: monitoring of distributed systems and access control system validation.

5.1 Access-Control

In the last decades a large variety of policies and policy models have been proposed to define authorized operations on specific resources. Well known examples of access control models include Mandatory Access Control (MAC) Model, Discretionary Access Control (DAC) Model, Role-Based Access Control (RBAC) Model and more recently eXtensible Access Control Markup Language (XACML) conforming services and applications. Many framework have been presented for specification and validation of access control policies [13–15]. Some works such as [13] use UML based models for expressing role based access control on domain concepts and provide automated facilities for translating these models into XACML code. Other works extract information from business process models to formulate a set of XACML based security policies [16] or derive process-related RBAC models from process execution histories [17]. These models do not consider policy detection and management mechanisms in large and complex environments able to detect inconsistencies of policy specifications and allow policy adaptability to evolving contextual behaviour or technological environment. Some proposals representing an attempt to address this issue are the works in [6,18]. Specifically, the authors of [6] present a dynamic policy management process and access management system that integrates the analysis of user management data as well as contextual data. Similarly to our approach, this work addresses policy recommendations based on contextual data and KPI validation but it relies on mining engine more than on adaptable and flexible monitoring facilities. The work in [18] presents a self adaptive authorization framework to automatically identifying security flaws and autonomously change the access control policy configuration. Similarly to our approach this framework uses monitoring facilities of the authorization system to identify abnormal behaviour and is compliant with RBAC/ABAC authorization infrastructure. Differently from [18], the main goal of the policy monitoring infrastructure proposed in this paper is not to implement self adaptation but providing feedbacks to policy validators for updating and refining policies according to contextual changes.

5.2 Monitoring

Several general-purpose monitoring proposals are currently available, which can be mainly divided into two groups: those that are embedded in the execution engine such as [19,20] and those that can be integrated into the execution framework as an additional component such for instance [21]. Both the solutions have specific advantages. For sure, an embedded solution reduces the performance delay of the execution framework, mainly in terms of interactions and communication time. Coverage indicators can be directly evaluated by the execution framework, which can also execute corrective actions in case of important deviations. The main disadvantage of these approaches is the lack of flexibility both in the data collection, coverage measures definition and language adopted. Usually, in these proposals all the interested parameters, have to be predefined and modeled directly into the execution engine, by means of specific editors, and are dependent on the notation used for the policy definition. Thus any change requires to redesign or improve the execution engine itself, preventing in such manner the possibility of dynamic modification.

Among the additional monitor facility in this paper we refer to the monitoring framework called Glimpse [12], which is extremely flexible and adaptable to various scenarios and SOA architecture patterns.

6 Discussion and Conclusions

In this paper we proposed a Policy Monitoring Infrastructure to dynamically validate and monitor the actual resources access and users behavior in order to update, correct or improve the policy specification in granting/denying the accesses. Through the proposal of this paper policy validators can have a better knowledge of the actual policy usage, define and on-line calculate possible key performance indicators (KPIs) useful to adjust the policies and detect possible policy vulnerabilities or outdated rules.

The results collected in the experiment demonstrated the feasibility of the proposal and highlighted some of the potentialities of the Policy Monitor Infrastructure. Indeed without such a feature most of the weaknesses evidenced in the experiment and suggestions for access control policy improvements were not easily identifiable.

In its simplicity the experiment open the path for further improvements of the proposed infrastructure. As future work we would like to integrate in the Infrastructure other coverage criteria, which can be more focused on the peculiarities of the access control system adopted in the different industrial context. We would like also to extend the Policy Monitoring Infrastructure considering different access and usage control policy specification languages. Finally we are planning to include in the Infrastructure a component able to automatically infer, from the coverage data collected, an alternative operational access control policy so to automatically validate and assess the original the XACML policy against the modification or changes experienced in the real world usage. This can

help the policy validators to get a clearer idea of the constraints and permissions expressed in the policy and identify the possible improvements or modifications.

Concerning threats to validity of the presented experiment, four aspects can be considered: the Rule Entities generation, user profiling for the simulation set up, the KPIs specification and the derivation of the XACML requests. Indeed, the coverage criterion, the sample of users selected during the interview stage and the KPIs defined may have influenced the reported results. Moreover due to manually derived XACML requests as well as the random selection of test cases in each observation windows, could be that different choices might have provided different effectiveness results. However the experiment would only be an example of Policy Monitoring Infrastructure usage.

Acknowledgments. This work has been partially supported by the GAUSS national research project (MIUR, PRIN 2015, Contract 2015KWREMX).

References

1. OASIS: Extensible Access Control Markup Language (XACML) version 2.0, February 2005
2. Martin, E., Xie, T., Yu, T.: Defining and measuring policy coverage in testing access control policies. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 139–158. Springer, Heidelberg (2006). https://doi.org/10.1007/11935308_11
3. Bertolino, A., Daoudagh, S., El Kateb, D., Henard, C., Le Traon, Y., Lonetti, F., Marchetti, E., Mouelhi, T., Papadakis, M.: Similarity testing for access control. Inf. Softw. Technol. **58**, 355–372 (2015)
4. Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Schilders, L.: Automated testing of extensible access control markup language-based access control systems. IET Softw. **7**(4), 203–212 (2013)
5. Calabrò, A., Lonetti, F., Marchetti, E.: Access control policy coverage assessment through monitoring. In: Proceedings of TELERISE 2017 Workshops, Trento, Italy, 12 September 2017 (2017, to apper)
6. Hummer, M., Kunz, M., Netter, M., Fuchs, L., Pernul, G.: Adaptive identity and access management–contextual data based policies. EURASIP J. Inf. Secur. **2016**(1), 19 (2016)
7. Rapps, S., Weyuker, E.: Selecting software test data using data flow information. IEEE Trans. Softw. Eng. **SE-11**(4), 367–375 (1985)
8. Zhu, H., Hall, P.A.V., May, J.H.R.: Software unit test coverage and adequacy. ACM Comput. Surv. **29**(4), 366–427 (1997)
9. Bertolino, A., Marchetti, E., Morichetta, A.: Adequate monitoring of service compositions. In: Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2013, pp. 59–69 (2013)
10. Bertolino, A., Calabró, A., Lonetti, F., Marchetti, E.: Towards business process execution adequacy criteria. In: Winkler, D., Biffl, S., Bergsmann, J. (eds.) SWQD 2016. LNBIP, vol. 238, pp. 37–48. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27033-3_3
11. Bertolino, A., Le Traon, Y., Lonetti, F., Marchetti, E., Mouelhi, T.: Coverage-based test cases selection for XACML policies. In: Proceedings of ICST Workshops, pp. 12–21 (2014)

12. Bertolino, A., Calabrò, A., Lonetti, F., Di Marco, A., Sabetta, A.: Towards a model-driven infrastructure for runtime monitoring. In: Troubitsyna, E.A. (ed.) SERENE 2011. LNCS, vol. 6968, pp. 130–144. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24124-6_13
13. Daoudagh, S., El Kateb, D., Lonetti, F., Marchetti, E., Mouelhi, T.: A toolchain for model-based design and testing of access control systems. In: 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 411–418. IEEE (2015)
14. Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E.: An automated testing framework of model-driven tools for XACML policy specification. In: 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 75–84. IEEE (2014)
15. Ferraiolo, D., Atluri, V., Gavrila, S.: The policy machine: a novel architecture and framework for access control policy specification and enforcement. J. Syst. Architect. **57**(4), 412–424 (2011)
16. Wolter, C., Schaad, A., Meinel, C.: Deriving XACML policies from business process models. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE 2007. LNCS, vol. 4832, pp. 142–153. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77010-7_15
17. Baumgrass, A., Schefer-Wenzl, S., Strembeck, M.: Deriving process-related RBAC models from process execution histories. In: 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, pp. 421–426, July 2012
18. Bailey, C., Chadwick, D.W., De Lemos, R.: Self-adaptive authorization framework for policy based RBAC/ABAC models. In: 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 37–44. IEEE (2011)
19. Daoudagh, S., Lonetti, F., Marchetti, E.: Assessment of access control systems using mutation testing. In: Proceedings of TELERISE, pp. 8–13 (2015)
20. Mouelhi, T., El Kateb, D., Le Traon, Y.: Chapter five-inroads in testing access control. Adv. Comput. **99**, 195–222 (2015)
21. Carvallo, P., Cavalli, A.R., Mallouli, W., Rios, E.: Multi-cloud applications security monitoring. In: Au, M.H.A., Castiglione, A., Choo, K.-K.R., Palmieri, F., Li, K.-C. (eds.) GPC 2017. LNCS, vol. 10232, pp. 748–758. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57186-7_54