Published in IET Software
Received on 25th June 2012
Revised on 22nd December 2012
Accepted on 27th February 2013
doi: 10.1049/iet-sen.2012.0101

ISSN 1751-8806

Automated testing of eXtensible Access Control Markup Language-based access control systems

Antonia Bertolino[1], Said Daoudagh[1], Francesca Lonetti[1], Eda Marchetti[1], Louis Schilders[2]

[1]Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR via G. Moruzzi 1, 56124 Pisa, Italy
[2]CEO, Custodix NV, 9830 Sint-Martens-Latem, Belgium
E-mail: francesca.lonetti@isti.cnr.it

Abstract: The trustworthiness of sensitive data needs to be guaranteed and testing is a common activity among privacy protection solutions, even if quite expensive. Accesses to data and resources are ruled by the policy decision point (PDP), which relies on the eXtensible Access Control Markup Language (XACML) standard language for specifying access rights. In this study, the authors propose a testing strategy for automatically deriving test requests from a XACML policy and describe their pilot experience in test automation using this strategy. Considering a real two-level PDP implemented for health data security, the authors compare the effectiveness of the test plan automatically derived with the one derived by a standard manual testing process.

1 Introduction

Nowadays assuring that detailed information can be securely exchanged between different platforms is a real and pressing need. In this regard, the European Project Trusted Architecture for Securely Shared Services (TAS3) [1] is developing a trusted architecture for the delivery of adaptive security services that preserve personal privacy and confidentiality in dynamic environments. One of the TAS3 pilot applications focuses on the interactions between patients and healthcare professionals, who should be able to access the repositories containing the relevant medical history of a patient at any time.

The trustworthiness of the TAS3 framework is strongly dependent on correct data protection and access control management, and consequently on an accurate testing phase of the implementation of the access control policies. In TAS3 the policies for access control decision are specified in eXtensible Access Control Markup Language (XACML) [2]. Therefore a good methodology for the derivation of test inputs (XACML requests) is needed. This could be used for probing the XACML policy implementation engine, called the policy decision point (PDP), and checking the PDPs responses against the expected ones. However, the generation of a good set of XACML requests revealed to be a difficult process and its manual management required a quite big effort. Owing to its past experience in using and developing XACML-based test strategies [3–5], the national research council (CNR) group to which the paper co-authors belong, collaborated with the Custodix [6] private company, specialised in data protection solutions for eHealth, for an effective improvement of the testing process.

Therefore a joint effort between Custodix and a CNR research group, both TAS3 partners, was launched for introducing automated tool support in the validation of XACML policy implementation engines. We report in this paper the experience gained during this joint work and the developed strategy.

The rest of this paper is structured as follows. Sections 2 and 3 provide the context in which the experience has been gained and a brief description of the XACML language, respectively. Section 4 presents the standard test process adopted by Custodix. We then introduce a new test strategy and its comparison with an existing one in Section 5. The application of the proposed strategy within Custodix testing scenario is shown in Section 6. Further considerations about manual and automated testing are discussed in Section 7, whereas experience feedbacks are provided in Section 8. Finally, Section 9 puts our work in context of related work and Section 10 draws conclusions.

2 Testing scenario

One of the main aims of the TAS3 project is around the secure management of patients data inside the healthcare applications.

In particular, Custodix [6] has developed patient information location service (PILS), an engine supporting the distributed search and recovery of patient information from different sources. PILS users need to authenticate themselves using their electronic identity card (the Belgian eID) and PILS access policies are used to rule access to data. By the provided graphical user interface (GUI), the users can themselves set the policies indicating which information they want to share with which healthcare professionals.

Fig. 1 shows a simplified version of the testing scenario, considered inside the TAS3 project, where a generic web user access to the portal is exemplified. In particular, the

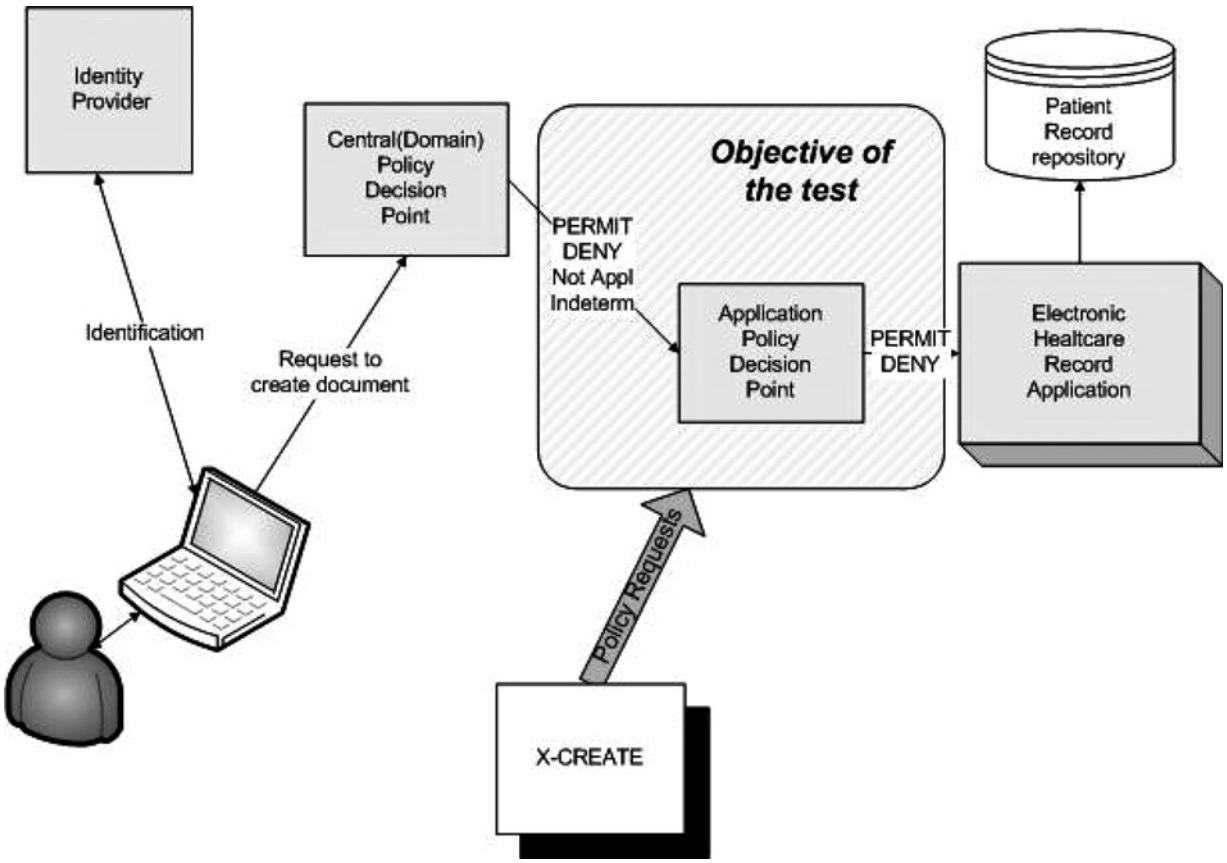

Fig. 1 *Testing scenario*

PILS portal creates an identification request for the identity provider (IdP), that contains the authentication method chosen by the user. The IdP intercepts this request, inspects it to check the user credentials and creates a response. At this point the user can either create the necessary access policy for a defined type of healthcare professional (pharmacist, general practitioner, specialist) or control/modify his/her personal preferences regarding data protection or check the audit logs of who accessed his/her data. In TAS3 the access management has been implemented by a distributed access control framework. In the pilot application in which Custodix was involved, two levels of PDP are implemented: the central (domain) policy decision point (Central PDP) and the application policy decision point (Application PDP).

The Central PDP checks the creation or modification of a policy by the patient. This policy rules the reading, creation or deletion of the patient record or information for a defined type of healthcare professional and is processed on the Application PDP that checks the access of the healthcare professional to the patient data. As evidenced by Fig. 1, the objective of testing was the Application PDP and the test plan was obtained by an automatic support tool, called XaCml REquests derivAtion for TEsting (X-CREATE).

3 XACML

XACML [2] is a platform-independent extensible markup language (XML)-based language for the specification of access control policies. A policy consists of a target, a set of rules and a rule-combining algorithm. The target specifies the subjects, resources, actions and environments on which a policy can be applied. Each subject, resource, action and environment contains two main attributes that are `<ATTRIBUTEID>` and `<DATATYPE>` and an `<ATTRIBUTEVALUE>` element that specifies the associated value. If a request satisfies the target of the policy, then the set of rules of the policy is checked, else the policy is skipped. The rule is composed by a target, which specifies the constraints of the requests to which the rule is applicable. The rule has a condition which is a boolean function evaluated when the rule is applicable to a request. If the condition is evaluated to true, the result of the rule evaluation is the rule effect (‘Permit’ or ‘Deny’), otherwise a ‘NotApplicable’ result is given. If an error occurs during the application of a policy to the request, ‘Indeterminate’ is returned. The rule-combining-algorithm specifies the approach to be adopted to compute the decision result of a policy when more than one rule may be applicable to a given request. For instance, the ‘permit-overrides’ algorithm specifies that ‘Permit’ takes the precedence regardless of the result of evaluating any of the other rules in the policy, then it returns ‘Permit’ if there is a rule, that is evaluated to ‘Permit’, otherwise it returns ‘Deny’ if there is at least a rule, that is evaluated to ‘Deny’ and all other rules are evaluated to ‘NotApplicable’. If there is an error in the evaluation of a rule with ‘Permit’ effect and the other policy rules with ‘Permit’ effect are not applicable, the ‘Indeterminate’ result is given.

The access decision is given by considering all attribute and element values describing the subject, resource, action and environment of an access request and comparing them with the attribute and element values of a policy.

doi: 10.1049/iet-sen.2012.0101

4 Manual testing of healthcare policies

Notwithstanding Custodix experience, the testing of the XACML PDP remained an error-prone and effort consuming activity, because XACML is not conceived for human manipulation. This implies the necessity of automation, so to drastically decrease the testing time and effort. For testing the two-levels PDP presented in Section 2, Custodix policies were: 'create-document', 'read-document', 'delete-document', 'read-information-unit', 'read-patient, dashboard', that specify the creation of a document in a patient record, the reading of a document, the deletion of a document from a patient record, the reading of a document content, the verification whether the patient has documents in the repository, the access for setting and changing a policy, respectively. Table 1 provides the number of rules, conditions, subjects, resources, actions and functions (columns 1, 2, 3, 4, 5, 6, respectively) characterising the considered policies.

For each of these policies, a test suite was manually derived and executed on the Application PDP and results were collected. For this activity Custodix did not follow a systematic approach; test requests were *ad-hoc* derived either exploiting the domain knowledge and the expertise of the involved personnel, or randomly generated using the values of the policy itself. Approximately, for each policy of Table 1 a test plan of around ten test requests was derived with the only exception of the 'dashboard' for which 30 test requests were considered. Each test request was then processed on the Application PDP. The overall testing process, from the test derivation to the test execution, took three working days.

Test results evidenced that all requests manually derived were successfully executed, which led Custodix to the consideration that the Application PDP correctly managed the users access.

5 X-CREATE test strategies

In the previous works [3–5] we addressed the automatic derivation of XACML requests proposing three main testing strategies based on a combinatorial approach of the values taken from the policy. Specifically, we defined: the 'Simple Combinatorial' testing strategy [4] that derives a request for each simple combination of the policy values; the 'XPT-based' testing strategy [3] that generates requests using the structures obtained applying the 'XPT' strategy [7] to the XACML Context Schema [2], and finally an improvement of the 'XPT-based' strategy, called 'Incremental XPT' [4], able to reduce the number of generated requests.

Analysing the previous proposed strategies [4, 5], we noticed an impact of the policy specification on the effectiveness of the derived test suites. In particular, the 'Simple Combinatorial' was not able to detect situations where the satisfiability of the policy rules depends simultaneously on the values of more than one subject, resource, action or environment. The 'XPT-based' strategy and the 'Incremental XPT' have a limited variability in the request structures used for test case derivation, this not always satisfies the complexity of the policy structure.

In this section we present a new test strategy for the automatic derivation of XACML requests, called 'Multiple Combinatorial', which overcomes the limitations of the already presented strategies.

We compare the fault-detection effectiveness of the 'Multiple Combinatorial' with that of 'Incremental XPT' strategy, which is currently the one that has been proved to be more effective and performing with respect to the different proposals [3–5, 8]. Both the strategies have been implemented into the X-CREATE tool [http://labse.isti.cnr.it/tools/xcreate] [4, 5]. Details about the two test strategies and their comparison are summarised in the next sections.

5.1 Multiple combinatorial

The 'Multiple Combinatorial' strategy relies on a combinatorial approach [9]. In particular, for each policy, four sets are generated, the 'SubjectSet', 'ResourceSet', 'ActionSet' and 'EnvironmentSet', containing the values of elements and attributes of the subjects, resources, actions and environments, respectively. Specifically, a subject entity is a combination of the values of `<ATTRIBUTEID>` and `<DATATYPE>` attributes and the value of the `<ATTRIBUTEVALUE>` element of the 'SubjectSet' set. Resource, action, and environment entities are similarly derived considering the 'ResourceSet', 'ActionSet' and 'EnvironmentSet' values. Random entities are also included in each set so that the resulting test plan could also be used for robustness and negative testing purposes. For instance, in the 'ResourceSet' a random entity for the resource is one having a random value (not specified in the policy) for the `<ATTRIBUTEVALUE>` element. An example of random value is `R:[B@1af7c57:3` of line 5 in Table 4.

We define for each set $S \in$ {SubjectSet, ResourceSet, ActionSet, EnvironmentSet}:

- The power set of S, called $P(S)$, as the set of all possible subsets of S.
- The cardinality of $P(S)$ as $\#P(S) = 2^n$, where n is the cardinality of S.
- The 'subset entity' as each element in $P(S)$. For instance, the element is called 'subject subset' if $S =$ SubjectSet.

The possible exponential cardinality of $P(S)$ is reduced by fixing the number of its subset entities. Indeed the necessary condition for a XACML request to be applicable on a field of the XACML policy (rule, target, condition) is that this request

Table 1 Healtcare policies data and test sets

#Rule	#Cond.	#Sub.	#Res.	#Act	#Funct.	Policy	#Req. by X-CREATE	#Req. additional	#Reduced test suite
3	2	1	2	1	3	create-document	24	2	14
4	3	2	4	1	3	read-document	90	4	22
3	2	1	3	1	3	delete-document	40	3	14
2	1	0	2	1	2	read-information-unit	12	1	10
4	3	2	4	1	3	read-patient	90	4	21
6	5	3	3	0	4	dashboard	300	6	37

Table 2 Mutation operators [13]

ID	Description
PSTT	policy set target true
PSTF	policy set target false
PTT	policy target true
PTF	policy target false
RTT	rule target true
RTF	rule target false
RCT	rule condition true
RCF	rule condition false
CPC	change policy combining algorithm
CRC	change rule-combining-algorithm
CRE	change rule effect

simultaneously includes all the entities that are specified in that policy field. Thus the XACML policy provides the minimum and maximum number of entities of the same type that have to be included in a request. For instance, if in a XACML policy there is never a condition or a target in which not less than two and not more than three subject entities are required for its evaluation, the minimum and maximum number of subject entities is 2 and 3, respectively. We use these numbers to (optionally) decrease the subject subsets.

The test requests are then generated by combining the subject, resource, action and environment subsets as in the following:

- apply the pairwise combination to cover all pairs (a, b) where $a \in A$, $b \in B$ such that A, $B \in \{P(\text{SubjectSet}), P(\text{ResourceSet}), P(\text{ActionSet}), P(\text{EnvironmentSet})\}$ and $A \neq B$, we obtain the PW set;
- similarly apply the threewise, to cover all triples (a, b, c) where $a \in A$, $b \in B$ and $c \in C$, such that A, B, $C \in \{P(\text{SubjectSet}), P(\text{ResourceSet}), P(\text{ActionSet}), P(\text{EnvironmentSet})\}$ and $A \neq B \neq C$, we obtain the TW set;
- apply the fourwise, that is all possible combinations of the subject subsets, resource subsets, action subsets and environment subsets, we obtain the FW set.

Because the inclusion property is $\text{PW} \subseteq \text{TW} \subseteq \text{FW}$, duplicated combinations have been eliminated considering the following sets: PW called 'Pairwise', TW\PW called 'Threewise' and FW\(TW ∪ PW) called 'Fourwise'.

Considering first Pairwise set, then Threewise set and finally Fourwise set, for each combination a XACML request containing the subset entities is generated. The maximum number of requests derived by this strategy is equal to the cardinality of FW set.

5.2 Incremental XPT

For aim of completeness, we shortly summarise the steps of the 'Incremental XPT' strategy referring [4] for further details. The strategy consists of three main steps:

1. *Intermediate-request generation:* Given the XACML Context Schema [2], a set of XML instances (729 structurally different intermediate requests) is generated by applying a variant of the Category Partition (CP) method [10] and traditional boundary conditions;
2. *Policy-under-test analysis:* As for 'Multiple Combinatorial' the sets 'SubjectSet', 'ResourceSet', 'ActionSet', 'EnvironmentSet' are defined from the analysis of the policy;
3. *Request values assignment:* The combinations of subject, resource, action and environment entities (for entities definition see Section 5.1) are taken one by one so to completely fill the set of the generated intermediate requests.

5.3 Comparison of test strategies

We compare 'Multiple Combinatorial' and 'Incremental XPT' strategies in terms of fault-detection capability. For the comparison we used (see Table 3) eight XACML policies (specifically demo-5, demo-11, demo-26 taken from the Open Source repository software Fedora [11], one of the repositories used within the TAS3 project, and five policies released for another pilot application of the TAS3 project).

To measure fault-detection, we applied mutation analysis [12], that is a standard technique to assess the effectiveness of a testing approach. The program under test is modified in order to produce a set of mutants, each containing one fault. Each test case is executed on the original program and its mutants, then outputs are collected: if the mutant's output is different from the original program's one, the fault is detected and the mutant is said to be killed.

We generated for each policy a set of mutants (see second column of Table 3) by considering the mutation operators for XACML policies defined in [13]. These operators, listed in Table 2, introduce syntactic faults, by mutating the predicates for the target and condition elements, and also emulate semantic faults, by modifying the logical constructs of the policies. We refer to [13] for their detailed description.

The requests obtained by the 'Multiple Combinatorial' and those obtained by 'Incremental XPT' have been executed on each policy and its mutants. The percentage of mutants killed by a set of test requests is taken as a measure of fault-detection effectiveness of the two strategies.

Table 3 Mutant-kill ratios achieved by test suites of 'Multiple Combinatorial' and 'Incremental XPT'

		Multiple Combinatorial				Incremental XPT			
		TSEff		TSDecr		TSEff		TSDecr	
Policy	# Mut.	# Req.	Mut. Kill, %	# Req.	Mut. Kill, %	# Req.	Mut. Kill, %	# Req.	Mut. Kill, %
demo-5	23	210	100	210	100	729	100	210	100
demo-11	22	100	100	100	100	729	100	100	100
demo-26	17	40	94.12	40	94.12	729	94.12	40	88.23
student-application-1	15	40	93.75	40	93.75	729	93.75	40	93.75
student-application-2	15	368	93.75	368	93.75	729	93.75	368	93.75
university-admin-1	20	5400	95.24	729	76.19	729	95.24	729	95.24
university-admin-2	20	5400	95.24	729	76.19	729	95.24	729	95.24
university-admin-3	20	4636	95.24	729	76.19	729	95.24	729	95.24

doi: 10.1049/iet-sen.2012.0101

Table 4 Test suite for 'create-document' policy and its PDP results

Row	Action	Resource-type	Central-pdp-decision	hcp-type	Application PDP result
1	create	documententry	Permit	physician	Permit
2	create	documententry	Permit	R:[B@1c634b9:2	Permit
3	create	documententry		physician	Permit
4	create	documententry		R:[B@1c634b9:2	Indeterminate
5	create	R:[B@1af7c57:3	Permit	physician	NotApplicable
6	create	R:[B@1af7c57:3	Permit	R:[B@1c634b9:2	NotApplicable
7	create	R:[B@1af7c57:3		physician	NotApplicable
8	create	R:[B@1af7c57:3		R:[B@1c634b9:2	NotApplicable
9	create	R:[B@1af7c57:3 documententry		physician	Permit
10	create	R:[B@1af7c57:3 documententry		R:[B@1c634b9:2	Indeterminate
11	create		Permit	physician	NotApplicable
12	create		Permit	R:[B@1c634b9:2	NotApplicable
13	R:[B@c5da6:2	documententry	Permit	physician	NotApplicable
14	R:[B@c5da6:2	documententry	Permit	R:[B@1c634b9:2	NotApplicable
15	R:[B@c5da6:2	documententry		physician	NotApplicable
16	R:[B@c5da6:2	documententry		R:[B@1c634b9:2	NotApplicable
17	R:[B@c5da6:2	documententry R:[B@1af7c57:3		physician	NotApplicable
18	R:[B@c5da6:2	R:[B@1af7c57:3	Permit	physician	NotApplicable
19	R:[B@c5da6:2	R:[B@1af7c57:3	Permit	R:[B@1c634b9:2	NotApplicable
20	R:[B@c5da6:2	R:[B@1af7c57:3		physician	NotApplicable
21	R:[B@c5da6:2	R:[B@1af7c57:3		R:[B@1c634b9:2	NotApplicable
22	R:[B@c5da6:2	R:[B@1af7c57:3 documententry		R:[B@1c634b9:2	NotApplicable
23	R:[B@c5da6:2		Permit	physician	NotApplicable
24	R:[B@c5da6:2		Permit	R:[B@1c634b9:2	NotApplicable
additional test cases					
25	create	documententry			Deny
26	create	documententry	Permit		Permit

In the first experiment, we generated for 'Multiple Combinatorial' and 'Incremental XPT' the maximum number of derivable requests. The third and fourth columns of Table 3 report the number of requests executed and the percentage of mutants detected using 'Multiple Combinatorial', whereas the seventh and eighth show the same data for the test suites derived with 'Incremental XPT'. From the obtained results we can conclude that the effectiveness of 'Multiple Combinatorial' strategy is generally equal to that of 'Incremental XPT'.

In the second experiment we considered the possibility of reducing the test suites evaluating the obtained fault detection. If the number of derivable requests of 'Multiple Combinatorial' is higher than 729, we limited this number to 729; otherwise if the number of requests of 'Multiple Combinatorial' is below 729 we adopt this number as a bound for 'Incremental XPT'. The fifth and sixth columns of Table 3 report the number of requests executed and the percentage of mutants detected using 'Multiple Combinatorial' whereas the ninth and tenth show the same data for the test suites derived with 'Incremental XPT'.

From the observed results we can draw the following conclusions:

- Limiting the 'Incremental XPT' to the 'Multiple Combinatorial' in general does not compromise the performance (Table 3 from the second to the sixth row). Only in case of 'demo-26' there is a loss in fault-detection effectiveness of the 'Incremental XPT': 'Multiple Combinatorial' reached 94.12 against 88.23 of the 'Incremental XPT'. In particular, the cardinality of the 'Multiple Combinatorial' test set could be considered as a good bound (stopping criterion) for the 'Incremental XPT'.
- Limiting the 'Multiple Combinatorial' to the 'Incremental XPT' in general compromises the performance (Table 3 from the seventh to the ninth row).

As general guideline the 'Multiple Combinatorial' strategy can be selected for test case generation without an evident loss in terms of fault-detection effectiveness when the cardinality of the derived test plan is lower than the maximum cardinality of the 'Incremental XPT' (i.e. 729).

For aim of completeness we point out that the X-CREATE tool, implementing the presented testing strategies, checks the correctness of the XACML policy file with respect to the XACML Policy Schema [2]. It is out of scope of the presented testing strategies dealing with the syntactic correctness of the values expressed in the policy. Thus we assume that the policy values have not syntactic errors.

6 Fully automated testing

In this section we describe the automatic test plan derivation for the Application PDP adopted by Custodix. In particular, CNR group finalised the definition of the automated testing process to be adopted inside Custodix in two rounds as described below.

6.1 First round: Test suites execution and results analysis

By means of X-CREATE, the selection guideline defined in Section 5.3 was applied to each policy of Table 1. For all policies, the 'Multiple Combinatorial' was the selected strategy and it was used for the derivation of the test set (Table 1 (eighth column) reports the cardinality of the test set for each policy). Custodix then executed each test set on the Application PDP and provided CNR group with the obtained results. The analysis of these data evidenced that:

- The X-CREATE test plan contained specific test requests useful for negative testing. These requests were derived by considering the subset entities including random values

(see Section 5). This kind of requests were only partially included in the manual test derivation and resulted useful for forcing the Application PDP to deal with not conventional/partially wrong access.

- Because the 'Multiple Combinatorial' test strategy took in consideration the combinations of all subset entities of the policy, the derived test requests provided a more specific and accurate test plan. Even when the execution of the requests did not reveal errors in the Application PDP, an increase in the confidence on the correct performance of this engine was experienced.

As an example, we show in Table 4 the results obtained by the execution of the test plan for the 'create-document' policy (see Listing 1). Specifically, the second column represents the <ATTRIBUTEVALUE> values associated to each <ATTRIBUTEID> contained in the 'ActionSet', the third and fourth those of the 'ResourceSet' (associated to 'resource-type' and 'central-pdp-decision' <ATTRIBUTEID>, respectively) and the fifth those of the 'SubjectSet'. The random values are those specified with the prefix 'R:[B@'. The last column contains the evaluation of the requests on the Application PDP.

```
<Policy PolicyId="urn:custodix:pils:policies:1"
 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
     algorithm:permit-overrides">
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
         ">
      documententry
     </AttributeValue>
     <ResourceAttributeDesignator AttributeId="resource-type"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">create</
          AttributeValue>
     <ActionAttributeDesignator AttributeId="action"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ActionMatch>
   </Action>
  </Actions>
 </Target>
 <Rule RuleId="DenyByDefault" Effect="Deny"> </Rule>
<Rule RuleId="pdpDecision" Effect="Permit">
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
       equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >Permit</AttributeValue>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
        one-and-only">
     <ResourceAttributeDesignator AttributeId="central-pdp-decision"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
   </Apply>
  </Condition>
 </Rule>
 <Rule RuleId="SenderIsPhysician" Effect="Permit">
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-
       in">
    <AttributeValue
     DataType="http://www.w3.org/2001/XMLSchema#string">physician</
         AttributeValue>
    <SubjectAttributeDesignator
     DataType="http://www.w3.org/2001/XMLSchema#string"
     AttributeId="attribute:hcp-type"/>
   </Apply>
  </Condition>
 </Rule>
</Policy>
```

Listing 1 *Create-document policy*

doi: 10.1049/iet-sen.2012.0101

```
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os␣
     access_control-xacml-2.0-context-schema-os.xsd">
 <Subject>
  <Attribute AttributeId="hcp-type"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>R:[B@1c634b9:2</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <Attribute AttributeId="resource-type"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>documententry</AttributeValue>
  </Attribute>
  <Attribute AttributeId="resource-type"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>R:[B@1af7c57:3</AttributeValue>
  </Attribute>
 </Resource>
 <Action>
  <Attribute AttributeId="action"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>create</AttributeValue>
  </Attribute>
 </Action>
 <Environment/>
</Request>
```

Listing 2 *A request for create-document policy*

Once executed the test plan of each policy, the CNR group further investigated if the generated XACML requests could cover all the policy verdicts so to guarantee a complete testing of all the possible behaviours of the Application PDP. This analysis evidenced that for 'create-document', 'read-document', 'delete-document', 'read-information-unit' and 'read-patient policies' the 'Deny' verdict was never covered. In addition, some of the test requests executions provided an 'Indeterminate' result. In particular, for the 'create-document' policy (in Listing 1), executing the 24 test cases derived by the 'Multiple Combinatorial' strategy, the obtained results were: 'four Permit', 'two Indeterminate' and '18 NotApplicable' as shown in Table 4. However, in Listing 1 the 'DenyByDefault' rule (line 28) has a 'Deny' effect that was not covered by the test suite.

Moreover, the PDP result associated to the request of Listing 2 (see tenth row of Table 4) is associated with an 'Indeterminate' result [The given error message is 'urn: oasis:names:tc:xacml:1.0:function:string-one-and-only expects a bag that contains a single element, got a bag with 0 elements'.]. This is because the evaluation of the condition function (line 33 of Listing 1) of the 'pdpDecision' rule requires that in the request there is a resource with an <ATTRIBUTEID> equal to 'central-pdp-decision' and that the associated <ATTRIBUTEVALUE> must be unique. However, the request of Listing 2 does not satisfy this constraint. The 'Indeterminate' result in this case is because of the semantics of the 'permit-overrides' rule-combining-algorithm (see details about the rule-combining-algorithms in Section 3).

The combinatorial approach implemented in 'Multiple Combinatorial' does not take into account the semantics of the rule-combining-algorithm and that of the XACML condition functions. Moreover, the 'Multiple Combinatorial' strategy includes only one random entity where the random value is associated to a randomly chosen <ATTRIBUTEID> (details about entities definition are specified in Section 5). For instance, for the 'create-document' policy the R:[B@1af7c57:3 random value has been associated to the 'resource-type' <ATTRIBUTEID> and not to the 'central-pdp-decision' <ATTRIBUTEID>. Then, the only entity containing the 'central-pdp-decision' <ATTRIBUTEID> has the 'Permit' value (see column fourth of Table 4). For this, the obtained test suite does not contain a request that makes false the condition of the 'pdpDecision' rule and then gives the possibility to obtain the 'Deny' result according to the semantics of the 'permit-overrides' rule-combining-algorithm and the result of the evaluation of the other policy rules.

6.2 Second round: Increasing test suites

To guarantee the coverage of all possible policy verdicts we increased the test suites derived by the 'Multiple Combinatorial' strategy with additional test cases. These test cases are able to guarantee decision coverage for the policy conditions. Specifically, following the Category Partition approach [10] we add the test cases that return a TRUE and FALSE evaluation of each rule condition, if they are not already included into the test set. In ninth column of Table 1 we report the number of additional test cases for each policy.

We executed the improved test suites derived from all policies of Table 1 on the Application PDP and verified that for each policy the 'Deny' verdict was covered. For instance, for the 'create-document' policy we added two requests (25th and 26th rows of Table 4), the one that gives a 'Deny' result is shown in Listings 3.

7 Manual against automated testing

The comparison between the manual and the automated test plan has been carried out considering the following aspects:

- The time necessary for test plan derivation and execution.
- The cardinality of the test suite.

```
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os␣
     access_control-xacml-2.0-context-schema-os.xsd">
 <Subject>
  <Attribute AttributeId="hcp-type"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue></AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <Attribute AttributeId="resource-type"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>documententry</AttributeValue>
  </Attribute>
  <Attribute AttributeId="central-pdp-decision"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue></AttributeValue>
  </Attribute>
 </Resource>
 <Action>
  <Attribute AttributeId="action"
   DataType="http://www.w3.org/2001/XMLSchema#string">
   <AttributeValue>create</AttributeValue>
  </Attribute>
 </Action>
 <Environment/>
</Request>
```

Listing 3 *An additional request for create-document policy*

- The effectiveness in finding problems or errors in the Application PDP.
- The coverage in terms of PDP verdicts.

The adoption of a tool as X-CREATE provided important savings in terms of effort and time for Custodix: manual test plan took three working days whereas the automated derivation of test suites took less than 20 min (for test cases derivation and execution). This is even more evident considering the cardinality of the test requests in each test plan. The manual test plan included around 80 test cases whereas X-CREATE around 570, thus a test plan seven times bigger in number is executed in negligible time. Note that the test plan analysis revealed that all test cases manually generated by Custodix personnel were included in the requests generated by X-CREATE. However, the manual test plan revealed strong expertise of the developers and their profound knowledge of the application.

Considering the test effectiveness, the X-CREATE test plan revealed to be an extremely important contribution. Thanks to the execution of some of the X-CREATE requests, Custodix discovered some errors in the management of the access decision from the Application PDP. In particular, the Application PDP decision resulted in contrast with that provided by the Central PDP and the verdict of the Application PDP overwrote that of the Central PDP. This problem, which never emerged before with the manual test execution, represented a vulnerability of the two-levels PDP and has been promptly corrected by Custodix.

Finally, we analysed the coverage of the verdicts of the two test plans. The manual test suites did not guarantee the coverage of all policy verdicts because the 'Deny' and 'Indeterminate' results were never experienced. With the improvements described in Section 6.2 of the X-CREATE test strategy, the coverage of all policy verdicts is assured. This increased the confidence in the goodness of the X-CREATE test plan.

8 Experience feedback and X-CREATE improvement

The weakest point in the manual and automated test process remained the analysis of the verdicts provided by the Application PDP. Since Custodix did not have an automated facility (usually called oracle), this analysis has been performed manually. The estimation provided for this activity was an average of 2 min for each verdict. Thus, even if the test effectiveness was improved by the automatic test plan, the advantage in effort reduction was reduced: the complete test plan execution, from the test case generation to the verdicts analysis, took three and half days for the manual test plan and two and half days for the automatic one.

Custodix and the CNR group worked together to define a new testing approach, specifically conceived for reducing the X-CREATE test suites guaranteeing their effectiveness in terms of verdicts coverage. The approach has been developed exploiting the peculiarities of the healthcare testing scenario presented in Fig. 1. As evidenced in this figure, the test cases derived by X-CREATE are executed only on the Application PDP; the Central PDP behaviour is simulated with the value of the <ATTRIBUTEVALUE> associated to the 'central-pdp-decision' <ATTRIBUTEID> contained in the policies of Table 1. Specifically, if a request executed on the Application PDP is applicable to the policy target, this simulates that the same request has been executed also on the Central PDP and the corresponding result has been returned [This result is specified as the value associated to the 'central-pdp-decision' <ATTRIBUTEID> contained in the 'pdpDecision' rule of the healthcare policies.]. For instance, in the 'pdpDecision' rule of the policy in Listing 1, the value 'Permit' of the <ATTRIBUTEVALUE> represents the result of the Central PDP execution. If a request executed on the Application PDP is not applicable to the policy target, this simulates either that the execution of the Central PDP has been bypassed (for instance for an intrusion

doi: 10.1049/iet-sen.2012.0101

attack) or that an error occurred in the Central PDP execution. For instance, the requests containing values specified in rows from fifth to eighth in Table 4 give NotApplicable because all include values of 'resource-type' that do not satisfy the policy target. Considering this specific situation, the following reduction criteria have been identified:

- A set of requests is useful if it contains the only values of the policy target that make it Applicable combined with the values of the attributes of the policy rules. These requests allow for the testing of the Application PDP in all cases in which the Central PDP decision has been given and depending on the combination of the values of the attributes of the policy rules, the Central PDP decision could be overridden by that of the Application PDP.
- A set of requests is useful if it contains all the possible combinations of the policy target attributes that make the target NotApplicable. These requests allow for the testing of the Application PDP in all cases in which the Central PDP decision is wrong or has been bypassed.

Referring to the test suite of Table 4 we mark with grey colour the rows corresponding to the requests included in the reduced test suite. Thus all requests making the target Applicable are included in the reduced test suite (for instance the first four grey rows). Moreover, only one request containing each of the four possible combinations of values of 'resource-type' and 'action' attributes is included to make the policy target NotApplicable. Considering for instance, the requests in the rows from fifth to eighth, which have the same values for 'action' and 'resource-type' and the same Application PDP result, only the fifth is included in the reduced test suite.

The possibility of creating a reduced test suite has been added to X-CREATE to better satisfy the Custodix exigences. We show in the last column of Table 1 the cardinality of the reduced test suites.

After executing the reduced test suites we analysed the obtained results and we observed that:

- The effectiveness of the reduced test suites in terms of coverage verdicts is preserved.
- The cardinality of the reduced test suites for some policies (for instance the 'dashboard' policy in the last row of Table 1) is greatly decreased.
- The analysis time of the verdicts of all reduced test suites took around 3 h.

This positive result has persuaded Custodix to introduce X-CREATE in the two-levels PDP testing process reducing considerably the time required for providing access control management, and improving the competitiveness of Custodix in developing data protection solutions for eHealth domain.

For the CNR researchers, this experience highlighted several weaknesses of the tool when dealing with the semantics of the rule-combining-algorithms and that of the XACML functions contained in the policy conditions. Test cases generation strategies focused on purely combinatorial approaches miss the semantics of these functions and that of the combining-algorithms making not accurate the testing results. We improved X-CREATE with a functionality taking into account the semantics of the functions and the combining-algorithms defined in the healthcare policies.

9 Related work

Testing of PDP is a critical issue and the complexity of the XACML language specification prevents the manual specification of a set of test cases capable of covering all the possible interesting critical situations or faults. This implies the need of automated test cases generation.

Some existing approaches consider the policy values in the test cases derivation. In particular, [8] presents the Targen tool that derives the set of requests satisfying all the possible combinations of truth values of the attribute id-value pairs found in the subject, resource and action sections of each target included in the policy under test. A different approach is provided by Cirg [14] that is able to exploit change-impact analysis for test cases generation starting from policies specification. In particular, it integrates the Margrave tool [15] which performs change-impact analysis so to reach high policy structural coverage. The X-CREATE tool [3–5] exploits the potentiality of the XACML Context Schema defining the format of the test inputs, and also applies combinatorial approaches to the policy values. In [3] a comparison between X-CREATE and the tool Targen [8] has been performed in terms of fault-detection capability, and the obtained results showed that X-CREATE has a similar or superior fault-detection effectiveness, and yields a higher expressiveness, as it can generate requests showing higher structural variability. In [4, 5] we present the advantages in terms of fault-detection effectiveness of the testing strategies implemented into X-CREATE tool. Our proposal here consists of a new testing strategy and its application for testing the XACML PDP used by a real-world Trusted Service Provider in the healthcare domain.

The authors of [16] address testing of the XACML PDP by running different XACML implementations for the same test inputs and detecting not correctly implemented XACML functionalities when different outputs are observed. Different from our proposal, this approach randomly generates requests for a given policy and requires more PDP implementations for providing an oracle facility by means of a voting mechanism. Our focus is on test cases derivation for PDP testing and not on oracle definition. A different solution for testing a PDP is presented in [17] where the authors provide a fault model and a test strategy able to highlight the problems, vulnerabilities and faults that could occur during the PDP implementation. The authors also provide a testing framework for the automatic generation of a test suite that covers the fault model. This approach deals with a specific authorisation system supporting usage control and history-based control and is specifically designed for PolPA language.

Other approaches target the testing of XACML policy and are based on the representation of policy implied behaviour by means of models [18–21]. Usually, these approaches provide methodologies or tools for automatically generating abstract test cases that have to be then refined into concrete requests for being executed.

10 Conclusions

We have proposed a testing strategy implemented into the X-CREATE tool to automatically derive a set of requests starting from a XACML policy. The proposed strategy guarantees the derivation of a set of requests meaningfully distributed over the input domain. In the reported experience X-CREATE has been used for deriving test sets from Custodix health policies and for providing an

automated test plan for the two-levels PDP. This experience has been a win–win endeavor for both the involved industrials and the researchers. The noticeable improvements in terms of time and effectiveness of the testing phase have convinced Custodix of the opportunity to introduce the X-CREATE tool into their deployment workflow. On the other side, for the CNR researchers the experimentation of the tool with the Custodix health policies paved the way for many improvements of the tool. Based on the results, we intend to improve X-CREATE with more effective XACML test derivation strategies focused on the application scenario constraints.

11 References

1 TAS3 Project. Trusted Architecture for Securely Shared Services. http://www.tas3.eu/

2 OASIS: 'eXtensible Access Control Markup Language (XACML) Version 2.0, February 2005. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20

3 Bertolino, A., Lonetti, F., Marchetti, E.: 'Systematic XACML request generation for testing purposes'. Proc. SEAA, September 2010, pp. 3–11

4 Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E.: 'Automatic XACML requests generation for policy testing'. Proc. ICST, April 2012, pp. 842–849

5 Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E.: 'The X-CREATE framework: a comparison of XACML policy testing strategies'. Proc. WEBIST, April 2012, pp. 155–160

6 Custodix. https://www.custodix.com/

7 Bertolino, A., Gao, J., Marchetti, E., Polini, A.: 'Automatic test data generation for XML schema-based partition testing'. Proc. AST, May 2007, pp. 10–16

8 Martin, E.: 'Automated test generation for access control policies'. Proc. OOPSLA, October 2006, pp. 752–753

9 Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: 'The AETG system: an approach to testing based on combinatiorial design', *IEEE Trans. Softw. Eng.*, 1997, **23**, (7), pp. 437–444

10 Ostrand, T.J., Balcer, M.J.: 'The category-partition method for specifying and generating functional tests', *Commun. ACM*, 1988, **31**, (6), pp. 676–686

11 FedoraCommons. Fedora Commons Repository Software. http://www.fedora-commons.org/

12 DeMillo, R.A., Lipton, R.J., Sayward, F.G.: 'Hints on test data selection: help for the practicing programmer', *Computer*, 1978, **11**, (4), pp. 34–41

13 Martin, E., Xie, T.: 'A fault model and mutation testing of access control policies'. Proc. WWW, May 2007, pp. 667–676

14 Martin, E., Xie, T.: 'Automated test generation for access control policies via change-impact analysis'. Proc. SESS, May 2007, pp. 5–12

15 Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C.: 'Verification and change-impact analysis of access-control policies'. Proc. ICSE, May 2005, pp. 196–205

16 Li, N., Hwang, J., Xie, T.: 'Multiple-implementation testing for XACML implementations'. Proc. TAV-WEB, July 2008, pp. 27–33

17 Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Martinelli, F., Mori, P.: 'Testing of PolPA authorization systems'. Proc. AST, June 2012, pp. 8–14

18 Le Traon, Y., Mouelhi, T., Baudry, B.: 'Testing security policies: going beyond functional testing'. Proc. ISSRE, November 2007, pp. 93–102

19 Mallouli, W., Orset, J.-M., Cavalli, A., Cuppens, N., Cuppens, F.: 'A formal approach for testing security rules'. Proc. SACMAT, June 2007, pp. 127–132

20 Li, K., Mounier, L., Groz, R.: 'Test generation from security policies specified in or-BAC'. Proc. COMPSAC, July 2007, pp. 255–260

21 Pretschner, A., Mouelhi, T., Le Traon, Y.: 'Model-based tests for access control policies'. Proc. ICST, April 2008, pp. 338–347

doi: 10.1049/iet-sen.2012.0101