Assessing Testing Strategies for Access Control Systems: A Controlled Experiment

Said Daoudagh[1,2][a], Francesca Lonetti[1][b] and Eda Marchetti[1][c]
[1]*ISTI-CNR, Pisa, Italy*
[2]*Department of Computer Science, University of Pisa, Pisa, Italy*

Keywords: Access Control Systems, Controlled Experiment, Testing.

Abstract: This paper presents a Controlled Experiment (CE) for assessing testing strategies in the context of Access Control (AC); more precisely, the CE is performed by considering the AC Systems (ACSs) based on the XACML Standard. We formalized the goal of the CE, and we assessed two available test cases generation strategies in terms of three metrics: Effectiveness, Size and Average Percentage Faults Detected (APFD). The experiment operation is described and the main results are analyzed.

1 INTRODUCTION

Access Control Systems (ACSs) play a critical role inside the information technology systems. They are concerned with determining the allowed activities of legitimate users, ensuring that only the intended subjects can access the protected data and resources. Most of the existing access control systems rely on the eXtensible Access Control Markup Language (XACML) (OASIS, 2013), the de facto standard for the specification of policies and requests.

Testing of access control systems represents a key activity to guarantee the trustworthiness of sensitive data and protect information technology systems against inappropriate or undesired user access. Several strategies for the generation of test cases (i.e., access requests) for access control systems have been defined in scientific literature. They leverage the application of the combinatorial approaches to XACML policies values for generating test inputs (Bertolino et al., 2013b; Bertolino et al., 2012a; Martin and Xie, 2006); or exploit change-impact analysis for test cases generation starting from policies specification (Martin and Xie, 2007); or are based on the representation of policy-implied behavior by means of models (Mouelhi et al., 2008; Daoudagh et al., 2019b)

The need for effective and efficient evaluation methods of test cases generation strategies is growing in order to gain confidence that a system meets its security requirements. Indeed, the selection of the most effective test generation strategy for the access control systems allows: i) on one side to exercise all the security-critical aspects and discover all the possible faults of the access control systems; ii) on the other side to develop a successful and cost-effective testing phase. The assessment of the most effective test cases generation strategies for XACML based access control systems usually relies on several techniques, e.g., coverage (Martin et al., 2006) or mutation analysis (Bertolino et al., 2013a) and is based on specific metrics and evidences gathered or from formal assurance techniques or from experimental evaluations.

However, in software engineering there is still an important lack in the application of standardized and systematic procedures for performing assessment or generalization of case studies results. Usually, associating to the results the standard means or the deviation values as well as their confidence levels are the synonymous of statistical evidences. But, these values are meaningless if the testing process cannot be rigorously conducted and, more important, replicated. Indeed, software engineering research activities still miss the knowledge of what is a *Controlled Experiment* and how to apply it. Controlled experiments have a very long application history in disciplines as medicine, biology and chemistry, because they are the most effective means for minimizing the effect of variables which are not of interest of the study. The Controlled Experiment lets to focus on an object, population, or any other variable which a scientist would

[a] https://orcid.org/0000-0002-3073-6217
[b] https://orcid.org/0000-0002-4864-2219
[c] https://orcid.org/0000-0003-4223-8036

Daoudagh, S., Lonetti, F. and Marchetti, E.
Assessing Testing Strategies for Access Control Systems: A Controlled Experiment.
DOI: 10.5220/0008974201070118
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 107-118
ISBN: 978-989-758-399-5; ISSN: 2184-4356

like to control and fixes the exact conditions in which the experiment can be performed and replicated: no deviations in the environment of the experiment can influence the final results.

Even if well known Controlled Experiments are rarely applied in software engineering (Jedlitschka and Pfahl, 2005) and specifically in software testing (Do et al., 2005) with the aim of obtaining reliability as well as generalization of results, in the context of testing of XACML based access control systems there is a complete lack of such kind of well defined approaches.

Thus, the aim of this paper: to provide for the fist time a Controlled Experiment for evaluating two XACML based test cases generation strategies available in literature, specifically Multiple Combinatorial (Bertolino et al., 2013b) and XACMET (Daoudagh et al., 2019b). We use the Goal Question Metric (GQM) template (Basili and Rombach, 1988) to formalize the goal of the experiment and define three metrics that are: effectiveness, size and average percentage faults detected (APFD). Moreover, as infrastructure for performing the controlled experiment, our proposal leverages the XACML Mutation Framework (XMF) (Daoudagh et al., 2019a) that allows for replicability of the experiment as well as generalization of results, finding aggregating, and finally reducing of experimentation costs.

In summary, the main contribution of the paper is the formal definition of a Controlled Experiment within the access control domain. Therefore, according to (Wohlin et al., 2012; Juzgado and Moreno, 2001), we present the thee main steps:

- the definition of a controlled experiment for the evaluation of two test cases generation strategies;
- the instrumentation and execution of the experiment;
- the analysis of the results.

Outline. Section 2 introduces the basic concepts used in the remaining sections. We define goals, context and variables of our controlled experiment in Section 3. Then, Section 4 presents the execution of the experiment whereas Section 5 provides an analysis of the results. Section 6 presents related works. Finally, Section 7 shows discussions and conclusions.

2 BACKGROUND

Access Control. Access Control (AC) is a mechanism used to restrict access to data or systems according to Access Control Policy (ACP), i.e., a set of rules that specify who has access to which resources and under which circumstances (Sandhu and Samarati, 1994).

Among the AC models proposed in the literature, we refer to the Attribute-Based Access Control (ABAC) (Jin et al., 2012) which is currently one of the most adopted in industrial environment (Hu et al., 2019).

The basic idea of ABAC is to use attributes of different entities to formulate access control decisions regarding a subject's (e.g., user or process) access on an object (e.g., file or database) in a system. The authorization decisions are obtained by means of authorization policies specified using a policy specification language. ABAC policies are a set of rules defined based on the attributes of subjects, objects and operations as well as other attributes, such as contextual or environmental attributes.

The ABAC model is usually implemented using the XACML (OASIS, 2013) standard that defines ACPs and access control decision requests/responses in a XML format. A XACML policy defines the access control requirements of a protected system. An access request that aims at accessing a protected resource is first evaluated against the policy, after which the access is granted or denied. A *XACML Request* is composed of four main elements: 1. *Subject*, the entity requesting the access; 2. *Resource*, the requested object that is described in terms of attributes; 3. *Action*, the operation that the subject wants to perform; and 4. *Environment*, the contextual information such as the request time and the location. The core component of a *XACML Policy* is the *Rule*, which represents the basic enforceable element: it is composed of an *Effect* (Deny or Permit value); *Target* and *Condition* which defines the applicability of the rule. The rules are evaluated by the Policy Decision Point (PDP) which is the AC component in charge of formulating an authorization decision. In particular, the effect of the rule is returned by the PDP when the evaluation of a given request meets the constrains of its target and condition. For more details about XACML we refer to its specification (OASIS, 2013).

Testing of Access Control Mechanisms. A typical AC systems testing process, shown in Figure 1, consists of at least four main steps: (A) *test cases generation*; (B) *mutants generation*; (C) *test cases execution*; and finally, (D) *results analysis*.

The first step is related to the generation of test cases (step Ⓐ) for the aim of generating a test suite starting from a given ACP. Among the available test cases generation strategies we refer to two different approaches: (1) *Multiple Combinatorial* strategy provided by X-CREATE tool (Bertolino et al., 2013b) generates test inputs using combinatorial approach of the XACML policies values; and (2) *XACMET* (Bertolino et al., 2012b; Bertolino et al., 2018) strategy based on the expected behavior of an XACML-based PDP. The next step (Ⓑ in Figure 1) is related to the generation of PDP mutants; the mutated versions of the PDP can be generated by applying a set of mutation operators. The basic idea of mutation testing is to simulate typical programmer's mistakes, by seeding syntactic faults in the original program in order to produce a set of faulty programs, called mutants, each one containing one fault. The main purpose of mutation testing is to assess the adequacy of a test suite. Each test case is executed on the original program and its mutants (in our case the PDP and the its mutated versions), then outputs are collected (i.e., the authorization responses): 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. The mutation score is the ratio of the number of detected faults over the total number of seeded faults and indicates the effectiveness of the test suite.

The results of steps Ⓐ and Ⓑ are then used in the next phase (step Ⓒ in Figure 1), that allows the execution of test cases on the original PDP and on its mutated versions. Finally, the result of step Ⓒ is used in step Ⓓ of the testing process (see Figure 1).

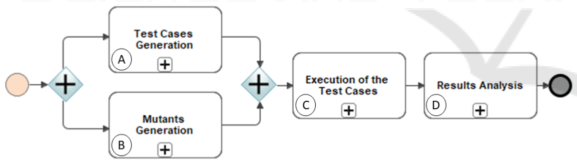

Figure 1: Workflow of the Testing Process.

All the above steps must be performed automatically to speed-up the testing phase. Among the automated frameworks for testing purpose of AC systems currently available, in this paper we refer to the XACML Mutation Framework (XMF) "which provides three main functionalities: test case generation, execution and assessment, and mutants generation" (Daoudagh et al., 2019a).

3 EXPERIMENTS DEFINITION AND PLANNING

The controlled experiment definition consists of three main steps: i) defining the goal of the experiment, the research questions and the associated Hypotheses that have to be formally tested as in Section 3.1; ii) introducing the context of the experiment as well as the variables, the subjects and the object of the experiment as in Section 3.2; iii) designing and instrumenting the experiment, i.e., the realization of the means for performing the experiment and monitoring it, without affecting the control of the experiment as in Section 3.3.

3.1 Goal and Hypotheses Formulation

According to the Goal Question Metric (GQM) template (Basili and Rombach, 1988), our research goal is as follows:

> ***Analyze*** two Test Generation Strategies (TGS_1 and TGS_2) ***for the purpose of*** evaluation ***with respect to their*** Effectiveness, Size and APFD of test suite produced ***from the point of view of the*** researcher ***in the context of*** XACML policy decision point testing.

In order to address the goal of our experiment, we defined three research questions and their associated hypotheses:

- **RQ1 Effectiveness:** How much does the quality of a test suite produced by $Strategy_1$ (TGS_1) differ from the quality of test suite produced by $Strategy_2$ (TGS_2) in terms of Effectiveness, i.e., the mutation score? For evaluating the effectiveness of the strategies we consider full test suites derived from each strategy;
- **RQ2 Size:** How much does the cost of a test suite produced by $Strategy_1$ differ from the cost of test suite produced by $Strategy_2$ in terms of Size, i.e., the number of test cases? For evaluating the cost of the strategies in terms of the number of XACML requests generated, we assume that all requests have the same cost in terms of generation and evaluation as well as verdict verification;
- **RQ3 APFD:** How much does the Average Percentage Faults Detected (APFD) of a test suite produced by $Strategy_1$ differ from the APFD of test suite produced by $Strategy_2$?

To answer the above research questions, the following Null Hypotheses have been defined:

- $H_{0Eff} : \mu_{EffTGS_1} = \mu_{EffTGS_2}$ the $Strategy_1$ finds on average the same number of faults, i.e., the effectiveness, as the $Strategy_2$, where μ denotes the average percentage of the killed mutants using the complete test suites generated by the two strategies;

- $H_{0Size} : \mu_{SizeTGS_1} = \mu_{SizeTGS_2}$ the size of test suite *is equal* for $Strategy_1$ and $Strategy_2$;
- $H_{0APFD} : \mu_{APFDTGS_1} = \mu_{APFDTGS_2}$ the average APFD *is equal* for $Strategy_1$ and $Strategy_2$.

A null hypothesis states that there are no real underlying trends or patterns in the experiment setting; the only reasons for differences in the observations are coincidental. This is the hypothesis that we want to reject with a higher significance. When the null hypothesis can be rejected with relatively high confidence, it is possible to formulate an alternative hypothesis, as follows:

- $H_{1Eff} = \neg H_{0Eff}$;
- $H_{1Size} = \neg H_{0Size}$;
- $H_{1APFD} = \neg H_{0APFD}$.

3.2 Context, Variables and Subjects Selection

In the context of Access Control Systems (ACSs), the aim of our controlled experiment is the evaluation of test cases generation strategies by means of the mutation analysis at the level of the Java based PDP engine. The comparison involves the effectiveness of the test suite generated by each strategy, the cost associated to each test suite in terms of its size and the velocity at which that effectiveness is reached. According to the classification of the experiment context in (Wohlin et al., 2012), the comparison of the selected test strategies has been conducted through a *Multi-test within object study*, i.e., a kind of controlled experiment that examines a single object across a set of subjects. In this experiment the object is the PDP engine, while the subjects are the XACML policies.

The variables involved in the experiment are: one ***Independent variable***, i.e., the *test case generation strategy* with two levels or alternatives (treatments) for the main factor (i.e., Level 1 and Level 2) and three ***Dependent Variables***, i.e., the *Effectiveness*, the *Size* (or the *cost*) of the test suites and the *APFD* metrics. The ***Object of the experiment*** is the Policy Decision Point developed by Sun, SunPDP (Sun Microsystems, 2006).

According to (Juzgado and Moreno, 2001), the ***Parameter***, that could influence the result of the experiment or, alternatively, the response variable, is the *Mutation Generator* tool or strategy used to generate the mutated versions of the PDP. Among the currently available tools, we selected the μJava tool (Ma et al., 2006; seung Ma et al., 2005), but other solutions could be selected, such as Javalanche (Schuler and Zeller, 2009), Major (Just, 2014) or Judy (Madeyski and Radyk, 2010).

Finally, we defined the ***Subjects*** involved in the experiment, i.e., the XACML Policies. According to the recommendation in (Juzgado and Moreno, 2001), the selection is closely connected to the generalization of the results from the experiment, thus the selection must be representative for that population.

For this, we considered a set of real-world XACML policies taken from real contexts and European projects as summarized in Table 1. In particular, the columns represent the number of rules, conditions, subjects, resources, actions and distinct functions within each policy. As in the table, policies named demo-5, demo-11 and demo-26 have been taken from the Open Source repository software Fedora (Flexible Extensible Digital Object Repository Architecture) (fed, 2019) for controlling the access to the administered digital contents; the remaining six are those released by the TAS3 European project (TAS3 Project,).

The GQM template, considering the context of Access Control, is as the following:

> ***Analyze*** Multiple Combinatorial and XACMET Strategies ***for the purpose of*** evaluation ***with respect to their*** effectiveness and size of test suite produced ***from the point of view of the*** researcher ***in the context of*** XACML policy decision point testing.

3.3 Experiment Design and Instrumentation

The comparison between the performances of the different test strategies has been defined using the Paired Comparison design, a particular kind of one factor with two treatments (Wohlin et al., 2012) [1]. In this design, each subject uses both treatments on the same object. In the context of this paper it can be translated into: both test strategies (Multiple Combinatorial and XACMET) have to be applied to each XACML policy and both the obtained test suites evaluated using the SunPDP and its mutants.

According to (Wohlin et al., 2012), an important step for the controlled experiment is the instrumentation, i.e., the realization of the means for performing the experiment and monitoring it. In particular, the instruments considered are of three types, namely objects, guidelines and measurement instruments.

[1]The same design is called "randomized paired comparison design: two alternatives on one experimental unit" in (Juzgado and Moreno, 2001).

Table 1: XACML Policies Subjects.

Xacml Policy	Functionality					
	# Rule	# Cond	# Sub	# Res	# Act	# Funct
2_73020419964_2	6	5	3	3	0	4
create-document-policy	3	2	1	2	1	3
demo-5	3	2	2	3	2	4
demo-11	3	2	2	3	1	5
demo-26	2	1	1	3	1	4
read-document-policy	4	3	2	4	1	3
read-informationunit-policy	2	1	0	2	1	2
read-patient-policy	4	3	2	4	1	3
Xacml-Nottingham-Policy-1	3	0	24	3	3	2

Object. When planning for an experiment, it is important to choose objects that are appropriate. The object of our experiment is an XACML-based PDP named Sun PDP (Sun Microsystems, 2006), which is an open source implementation of the OASIS XACML standard, written in Java. We decided on Sun's PDP engine because it is currently one of the most mature and widely used engine for XACML policy evaluation, which provides complete support for all the mandatory features of XACML 2.0 as well as a number of optional features. This engine supports also all the standard attribute types, functions and combining algorithms and includes APIs for adding new functionalities as needed. The Sun PDP source code is broken into ten packages: seven packages include the core implementation, two packages include classes used for the configuration code, rarely used by programmers, and one package contains test code samples.

The comparison of the selected test strategies required the definition of the SunPDP Mutants. Therefore, through the Mutation Generator and Mutation Integrator components, the mutation operators (both class-level and method-level operators) have been applied to the Sun PDP code and executable mutant versions derived.

Guidelines and Measurement. Guidelines are needed to guide the participants in the experiment and they include process descriptions and checklists. Measurements are conducted via data collection that in human-intensive experiments are generally performed by manual forms or interviews. Since we conducted a technology-oriented experiment, we embedded both aspects in the automation process. Specifically, we leveraged the functionalities of XMF framework, i.e., test cases generation, mutants generation, test cases execution and results analysis, to perform our controlled experiment and automatically collect the data.

4 EXPERIMENT OPERATION

The experiment operation mainly consists of three steps: preparation, execution and data validation. Specifically, the preparation step focuses on the preparation of the subjects, object and the material needed.

In our experiment, the preparation step consists of: XACML policy selection, XACML requests generation, XACML based PDP selection, and finally the mutants generator selection.

During the execution step, the XACML requests are evaluated and obtained data are collected.

Finally, during the data validation step, the dependent variables are calculated, the collected data are managed and analyzed so to provide a valid picture of the experiment. In the following sections, we report the results of our data validation.

4.1 Data Validation

The data validation phase includes: i) a former descriptive statistics, which allows the visualization of the information using informal representations. In our experiment, these statistics span from the number of generated requests, to the distribution of the mutants on the different Java PDP classes, to the evaluation of the different test case executions. Even if simple and informal, these descriptive statistics let both to highlight an important criticality on the data set considered, that could have compromised the entire experiment, and to find out the corrective actions for solving it. Due to space limitation, in the following we report only the number of executions and the number of distinct mutants. ii) a latter formal description of the experiment results. In our experiment we focused on the Paired T-Test with Null Hypothesis and present the results in Section 5.

Number of Executions. Table 2 reports for each of the nine XACML policies, the number of executions for the test suites derived by the XACMET and the Multiple combinatorial test strategies (second and third column respectively). The total amount of executions is reported in the last column of the table. From the data collected, as reported in the last row, only 16% of the executions is performed by XACMET strategy, meaning that testing cost of such strategy is very low compared to the one of the Multiple Combinatorial strategy.

Table 2: Number of Executions by XACML Policy and Strategy.

Xacml Policy	# of Executions		All
	XACMET	Mulitlple	
2_73020419964_2	12320	481800	494120
create-document-policy	13560	96360	109920
demo-11	104390	321200	425590
demo-26	64240	128480	192720
demo-5	128480	674520	803000
read-document-policy	4014	240900	244914
read-informationunit-policy	27752	48180	75932
read-patient-policy	48180	240900	289080
Xacml-Nottingham-Policy-1	38892	55072	93964
All	441828	2287412	2729240

For aim of completeness, for each XACML policy, Figure 2 reports the percentage of executions for the two testing strategies considered. In particular, the blue bars (black in black and white printing) refer to the XACMET testing strategy, while the orange bars (light gray in black and white printing) report the percentage of executions of Multiple Combinatorial strategy.

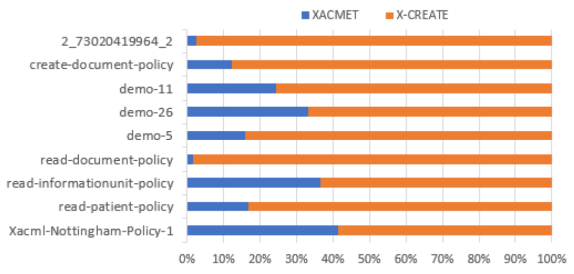

Figure 2: % of Executions by XACML policy, by XACMET and Multiple strategy.

Number of Distinct Mutants. The information about the number of executions can be analyzed also from the point of view of how many distinct mutated PDPs are evaluated by each XACML policy. In particular, Table 3 reports, in the second column, the number of distinct mutated PDPs considering the XACMET strategy, while in the third column those related to the Multiple Combinatorial ones are presented. By analyzing the data in the table only four of the nine XACML policies are evaluated by all mutated PDPs set. Specifically, for the first, second, sixth and seventh policies listed in the Table, only the test suite derived by the Multiple Combinatorial strategy is able to execute all the mutated PDPs; for the last policies neither of the two test suites is able to execute all the mutated PDPs.

Table 3: Number of Distinct MutatedPDP Evaluated by XACML Policy and Strategy.

Xacml Policy	# of Distinct MutatedPDP		All
	XACMET	Mulitlple	
2_73020419964_2	1540	8030	9570
create-document-policy	2712	8030	10742
demo-11	8030	8030	16060
demo-26	8030	8030	16060
demo-5	8030	8030	16060
read-document-policy	669	8030	8699
read-informationunit-policy	6938	8030	14968
read-patient-policy	8030	8030	16060
Xacml-Nottingham-Policy-1	1389	3442	4831
All	45368	67682	113050

The results reported in this table highlighted an important criticality for the evaluation of the controlled experiment metrics (Effectiveness and APFD). Indeed, because there is a difference in the number of distinct mutated PDPs for the two test suites, the Hypotheses testing could be invalidated and consequently also the answers to the target RQs. For a fair experiment and evaluation, it is important to guarantee that both strategies are evaluated using the same set of mutated PDPs. Therefore, the data sets have been reduced considering only the minimal common set of mutated PDPs for each test strategy.

As natural consequence such reduction has also an impact on the number of executions for the test suites derived by the XACMET and the Multiple combinatorial strategies as reported in Table 4. The comparison of the data of this table with those reported in the previous one (Table 2) shows that the reduction has the biggest impact on the Multiple Combinatorial strategy. For the sake of completeness we report, in Table 5, the percentage of reduction of executions associated to each XACML policy.

Table 4: Number of Reduced Executions by XACML Policy and Strategy.

Xacml Policy	# of Executions		All
	XACMET	Mulitlple	
2_73020419964_2	12320	92400	104720
create-document-policy	13560	32544	46104
demo-11	104390	321200	425590
demo-26	64240	128480	192720
demo-5	128480	674520	803000
read-document-policy	4014	20070	24084
read-informationunit-policy	27752	41628	69380
read-patient-policy	48180	240900	289080
Xacml-Nottingham-Policy-1	38892	22224	61116
All	441828	1573966	2015794

Table 5: % of Reduced Executions by XACML Policy and Strategy.

Xacml Policy	# of Distinct MutatedPDP		% of Reduction	
	XACMET	Mulitlple	Multiple	All
2_73020419964_2	1540	1540	81%	79%
create-document-policy	2712	2712	66%	58%
demo-11	8030	8030	0%	0%
demo-26	8030	8030	0%	0%
demo-5	8030	8030	0%	0%
read-document-policy	669	669	92%	90%
read-informationunit-policy	6938	6938	14%	9%
read-patient-policy	8030	8030	0%	0%
Xacml-Nottingham-Policy-1	1389	1389	60%	35%
ALL	45368	45368	31%	26%

5 RESULTS

According to (Juzgado and Moreno, 2001; Wohlin et al., 2012), we applied the Paired T-Test to formally verify the Null Hypothesis with the confidence level of 95%. This choice was a natural consequence of the type of design adopted, i.e, the paired comparison. Following the standard best practices, we decided to accept a probability of 5% of committing a Type-1-Error (Wohlin et al., 2012), i.e., the Null Hypothesis is rejected if the computed p-value is less or equal to 0,05 ($alpha = 0.05$).

Stating from the data collected we generated the necessary sample data so as to test each Null Hypothesis formulated in Section 3. In the next subsections detailed results for each research question are reported and discussed.

5.1 RQ 1: Effectiveness

As presented in Section 3 the aim of RQ 1 is to ***Analyze*** Multiple and XACMET Strategies ***for the purpose of*** evaluation ***with respect to their*** test suite effectiveness ***from the point of view of the*** researcher ***in the context of*** XACML PDP testing without constraints (i.e., considering the whole test suite generated).

A general attribute for the evaluation of the quality of a test cases generation strategy is its effectiveness, defined in terms of number (or percentage) of mutated PDPs killed. Therefore the effectiveness is calculated as:

$$Effectiveness = \frac{\#mutatedPDPsKilled}{\#mutatedPDPs}$$

It is important to remark that in order to correctly compute this measure the number of distinct killed mutants by each strategy and on each policy must be considered. The samples relative to the effectiveness of Multiple Combinatorial and XACMET test strategies for Null Hypothesis testing (H_{0Eff}) are reported in Table 6 (columns 2 and 3). In particular, the upper part of the table reports the samples associated to each test cases generation strategy, while the lower part reports some statistical information of each sample. Both the strategies have a similar behaviour: the test suites are able to kill less than 20% of the mutated PDPs.

Table 6: RQ 1: Effectiveness and RQ 2: Size.

Xacml Policies Subjects	Effectiveness		Size	
	Multiple	XACMET	Multiple	XACMET
2_73020419964_2	2,14	13,57	60	9
create-document-policy	15,30	14,93	16	5
demo-11	8,78	8,89	40	13
demo-26	7,67	8,97	16	8
demo-5	9,18	9,14	84	16
read-document-policy	19,88	19,43	30	6
read-informationunit-policy	8,59	8,91	6	4
read-patient-policy	7,68	8,79	30	6
Xacml-Nottingham-Policy-1	19,65	19,65	16	18
Samples Statistics				
N	9	9	9	9
Missing Count	0	0	0	0
Mean	10,987	12,4766	33,1111	9,4444
Standard Deviation	5,988	4,6064	24,9822	5,0525
Standard Error Mean	1,996	1,5355	8,3274	1,6841

This informal observation is confirmed also by the results of the Paired T-test associated to H_{0Eff} (Table 7, column 2).

Because the *p-value obtained* is $0,2705 > 0.05$, from the sample considered there is no difference from the point of view of effectiveness between the two test strategies.

For sure, the low values of fault detection effectiveness obtained in this experiment are not encouraging for any kind of test strategy. However from a deeper analysis we highlighted the two following main reasons.

First, the choice of XACML policies: these are real ones, therefore they contain the mostly used functionalities and constructs. They are not artificially developed for testing objective; therefore, they are not a complete representation of the XACML policy population. However, here the target is to asses the effectiveness of the test strategy on the few functions, data types, XACML elements that are currently used in the practice, so to focus as much as possible the testing activity on the most critical aspects.

The second reason concerns the mutation operators adopted. The operators implemented in the framework for the generation of mutated PDPs are the standard ones and are applicable to any kind of Java program. They do not consider the peculiarities of the XACML language and therefore could not be targeted by the XACML requests used as a tests input.

Thanks to this analysis we were able to discover two possible important improvements for the implemented testing framework as well as precious hints for future research activity in improving the mutation operators set and test strategy definitions.

5.2 RQ 2: Size

As presented in section 3, the aim of the RQ 2 is to ***Analyze*** Multiple and XACMET Strategies ***for the purpose of*** evaluation ***with respect to their*** cost in terms of number of test cases generated ***from the point of view of the*** researcher ***in the context of*** budget programming.

An important aspect for the selection of a test cases generation strategy is its cost evaluated in terms of: the time for the test cases execution and the time necessary for the debugging activity. Supposing that each test case has potentially the same impact on the overall testing effort, the execution time becomes directly connected with the number of test cases executed: i.e., the size of a test suite represents also its cost.

The samples relative to the sizes of Multiple Combinatorial and XACMET test strategies for Null Hypothesis testing (H_{0Size}) are reported in Table 6 (columns 4 and 5). In particular, the upper part of the table reports the samples associated to each test cases generation strategies, while the lower part reports some statistical information of each sample.

Except for *Xacml-Nottingham-Policy-1* policy, XACMET strategy generates smaller test suites with respect to those generated by Multiple combinatorial. Considering the size metric, the test suites of XACMET cost about 70% less than those of Multiple Combinatorial one, but reaching the same quality in terms of errors found or mutants killed.

This result is formally confirmed by the Paired T-test associated to H_{0Size} and illustrated in Table 7, column 3. The *p-value obtained* ($0{,}0253 < 0.05$) suggests, rejecting the Null Hypothesis H_{0Size}, of no difference from the point of view of the size between the two test strategies. Therefore we can conclude that the XACMET strategy outperformed Multiple strategy in terms of cost.

Table 7: Paired T-Test: RQ 1 (Effectiveness) and RQ 2(Size).

RQs	RQ 1	RQ 2
Label	Multiple - XACMET	
t	-1,1838	2,7426
df	8	8
p-value (2-tailed)	**0,2705**	**0,0253**
Mean	-1,4896	22,6667
Standard Deviation	3,7749	24,7942
Standard Error Mean	1,2583	8,2647
CI (Lower Bound)	-4,3913	3,6082
CI (Upper Bound)	1,412	41,7251

5.3 RQ 3: APFD

As presented in section 3 the aim of RQ 3 is to ***Analyze*** Multiple and XACMET Strategies ***for the purpose of*** evaluation ***with respect to their*** effectiveness in terms of APFD ***from the point of view of the*** researcher ***in the context of*** interruption of XACML PDP testing activity.

For unexpected time constraints or budgets reduction reasons, the testing activity could be interrupted before its overall completion. In this case, not all the test case could be executed with a consequent risk for the final quality and efficiency of the product developed.

In this case, the standard metrics adopted for measuring such a risk is the APFD metric, which is calculated by taking the weighted average of the percentage of faults detected during the execution of the test suite. It is formally defined as follows (Elbaum et al., 2002):

$$APFD = \frac{\sum_{i=1}^{n-1} F_i}{n \times l} + \frac{1}{2n}$$

where, n is the number of test cases in the test suite T, l is the number of faults, and F_i is the number of faults detected by at least one test case among the first i test cases in T.

By construction, APFD values range from 0 to 1; higher values imply faster (better) fault detection rates. Thus APFD is commonly used to evaluate the prioritization techniques because it is able to estimate the speed with which an ordered test suite can reach the maximum number of discovered faults. In other words, it is the measure of how quickly the faults are detected by a testing strategy. Improving the rate of fault detection can have an impact on the testing cost and effort: software engineers may locate and correcting faults earlier in advance and better evaluate the risk of test activity interruption (Elbaum et al., 2002; Bertolino et al., 2015).

Since the APFD measure is connected to the concept of (ordered) test suites having the same cardinality, its application in this experiment requires that the number of XACML requests generated by the Multiple Combinatorial and XACMET test strategy is the same. For this, the following corrective actions have been applied:

Test suite size and requests selection the cardinality N of the test suite has been fixed to the minimum one between the values of the Multiple Combinatorial and XACMET test strategy and the N test cases randomly selected among those available.

Prioritize the test cases each reduced test suite has been ordered in terms of mutated PDPs killed, so to assure the optimal fault detection effectiveness of each set.

APFD calculation for each ordered test suite the APFD has been calculated.

Considering in detail the *Prioritize the test cases* action, the technique used, called mutation-based heuristic, is a greedy-optimal selection of test cases computed on the mutation coverage. The heuristic is a sub-optimal algorithm, since it orders the test cases according to the cumulative number of different mutants killed. For implementing the selected heuristic the mutated PDPs killed by each test case have been organized into a matrix (request, mutated PDPs) where each cell(i, j) is equal to 1 if the i-th request kills the j-th mutated PDP, and 0 otherwise.

Informally, the algorithm for implementing the mutation-based heuristic works as follows:

1. it calculates the number of killed PDPs for each request;
2. it selects the request s.t. the number of killed PDPs is maximum;
3. it removes the request and the mutated PDPs killed by such request from the matrix;
4. it repeats the steps 1-3 until all requests are selected.

To avoid experimental bias, the above steps have been repeated ten times and the APFD computed.

The samples relative to the APFD of Multiple Combinatorial and XACMET test strategies for Null Hypothesis testing (H_{0APFD}) are reported in Table 8 (columns 2 and 3). In particular the upper part of the table reports the samples associated to each test case generation strategies, while the lower part reports some statistical information of each sample. The results of the Paired T-test associated to H_{0APFD} are illustrated in Table 9. Because the *p-value obtained* is

Table 8: RQ 3: APFD.

Xacml Policies Subjects	APFD	
	Multiple	XACMET
2_73020419964_2	0,932588598	0,9159689
create-document-policy	0,843173312	0,876790123
demo-11	0,956210668	0,953673777
demo-26	0,931931707	0,917708333
demo-5	0,964039803	0,965003406
read-document-policy	0,874445036	0,905128205
read-informationunit-policy	0,779981164	0,849919094
read-patient-policy	0,85009274	0,893767705
Xacml-Nottingham-Policy-1	0,980102041	0,978416605
Samples Statistics		
N	9	9
Mean	0,9014	0,9174
Standard Deviation	0,0676	0,0422
Standard Error Mean	0,0225	0,0141

Table 9: Paired T-Test: RQ 3(APFD).

RQs	RQ 3
Label	XACMET - Multiple
t	1,6135
df	8
p-value (2-tailed)	**0,1453**
Mean	0,016
Standard Deviation	0,0297
Standard Error Mean	0,0099
Confidence Interval Probability	0.95
Confidence Interval of the Difference (Lower Bound)	-0,0069
Confidence Interval of the Difference (Upper Bound)	0,0388

$0,1453 > 0.05$, from the sample considered there is no difference from the point of view of the APFD between the two test strategies, i.e., XACMET and Multiple Combinatorial have the same velocity in reaching their effectiveness.

6 RELATED WORK

The work presented in this paper spans over two main research directions: (1) XACML based test generation strategies and (2) modeling and supporting of controlled experiment in the context of access control systems.

XACML based Tests Generation. Considering the automated test cases generation, solutions have been proposed for testing either the XACML policy or the PDP implementation (Bertolino et al., 2014; Bertolino et al., 2013b). Among them, the most referred ones use combinatorial approaches for test cases generation. Specifically, the X-CREATE tool (Bertolino et al., 2013b) and the Targen tool (Martin and Xie, 2006) generate test inputs using combinatorial approaches of the XACML policies values and the truth values of independent clauses of policy values, respectively, whereas the work in (Pretschner et al., 2008) applies combinatorial analysis to the elements of the model (role names, permission names, context names) to derive test cases.

Alternatively, the Cirg approach (Martin and Xie, 2007) applies change impact analysis for test case generation starting from policy specification. Specifically, it provides a framework able to derive test cases as counterexamples that evidence semantic difference between two different versions of the policy under test.

Other approaches leverage existing symbolic execution techniques for generating test cases. Specifically, in (Li et al., 2014), first the access control policy under test is converted into semantically equivalent C Code Representation (CCR). Then, the CCR is symbolically executed to generate test inputs.

Differently from the above approaches, the work in (Daoudagh et al., 2019b) proposes the XACMET strategy based on the expected behavior of an XACML-based PDP. XACMET models the expected behaviour of the evaluation of a given XACML policy as a labeled graph and guarantees the full path coverage of such graph. The test cases generated are used only for the PDP testing purposes. The main benefits of XACMET deal with i) the derivation of XACML requests that explicitly take into account the semantics of XACML functions as well as the policy and rule combining algorithms; ii) the derivation for each test request of its expected verdict.

In the controlled experiment presented in this paper, we aim to compare the *Multiple Combinatorial* strategy implemented into X-CREATE tool (Bertolino et al., 2013b) and the XACMET strategy (Daoudagh et al., 2019b) in terms of effectiveness, size and APFD.

Supporting Controlled Experiment. Empirical validations play a key role in the evaluation of a software system. Validation in software engineering discipline, as in other research fields, relies on building different models of this discipline, i.e., modeling the objects of the domain, the processes manipulating the objects, the relations between processes and objects (Wohlin et al., 2012). The authors of (Wohlin et al., 2012) give first an overview of empirical strategies such as surveys, case studies, experiments and then define the main steps of experiment process such as scoping, planning, operation, analysis and interpretation, presentation and package.

The work in (Briand and Labiche, 2004) discusses specific challenges and issues of performing empirical studies with software testing techniques whereas the authors of (Do et al., 2005) identify two main complementary classes of empirical studies addressed in software testing: case studies (Runeson and Höst, 2008) and controlled experiments (Jedlitschka and Pfahl, 2005).

Controlled experimentation in software testing leverages numerous software artifacts, including for instance different versions of software systems, test suites, test object, fault data, mutated software. Obtaining such artifacts and organizing them in an environment able to support controlled experimentation is a difficult task.

The work in (Do et al., 2005; Do et al., 2004) presents first a survey of papers on testing that provide empirical studies identifying the main challenges of experimentation in software testing. These identified challenges are then used for designing and constructing an extensible infrastructure able to support controlled experimentation with software testing and regression testing techniques. This infrastructure provides guidelines for object selection, organization, and setup processes with the aim of reducing the costs of executing and replicating controlled experiments as well as aggregating results across experiments.

In this paper, we aim to leverage the advantages of the controlled experiments in the context of access control by defining and executing a controlled experiment for the evaluation of two test cases generation strategies adopted in literature.

7 DISCUSSION AND CONCLUSIONS

Automation and replication are two important aspects for the assessment of the effectiveness of different testing strategies and key factors for the overall testing process. Thus, the target of the paper: defining and executing a controlled experiment for the comparison of two test cases generation strategies, specifically *Multiple Combinatorial* and XACMET approach, in the context of access control.

According to the analysis and results in the presented experiment, the two strategies have the same effectiveness (RQ1) and behaviour in terms of APFD (RQ3), while they are significantly different in terms of cardinality of the test suites (RQ2), and therefore in their cost. This means that in case of budgets or effort constraints, because both of the test strategies provided the same performance in terms of fault detection, applying the XACMET approach could be a winning solution for reducing testing time and guaranteeing the same product quality.

A crucial aspect emerged during the comparative experiment analysis, that has never been considered before in the literature related to the assessment of XACML based systems: during the test suite execution, there is the possibility that different sets of distinct Mutated PDPs can be executed by different test strategies. Ignoring such data evidence and comparing the test strategies just in terms of effectiveness or APFD could really produce invalid conclusions and therefore wrong test strategy selection. By leveraging XMF and the data collected during the experiment execution, the mitigation of such a risk was possible and a correct evaluation provided.

Concerning the validity of the experiment, i.e., the amount of confidence in the results, the important key factors are Subjects, Object and Parameters of the experiment. Here below the strategies used to minimize the threats to validity are described.

With respect to *confidence in the results*, the controlled experiment used data and measurements that satisfy the principles of independence, homogeneity and normality.

With respect to *internal validity* the crucial points are: the policy used, the PDP engine selected and the tool integrated for the generation of the mutants set. Concerning the Subject of the experiment, the policies included in the XMF framework are a good representative of real world XACML Policies, because they contain most of the constructs and functionalities actually used in the practice. However different policies, like for instance those of XACML conformance test suite, may produce different results. The Object of the experiment, i.e., the Sun PDP implementation integrated in the XMF framework, is one of the most adopted in access control systems and therefore its quality and performance are well established. However, different XACML-based implementations could be considered. Finally, considering the Parameter of the experiment, i.e., the tool integrated for the derivation of the mutants set, we included the already existing μJava tool because it is one of the most widely used in object oriented environment. Previous works guarantee that its performance can be comparable to others available, however it could be possible that other mutation tools may produce different results.

With respect to *external validity*, we compared two test strategies: *Multiple Combinatorial* test strategy, provided by X-CREATE, which is one of the most referred in the context of access control systems, and an innovative one, the XACMET test strategy, so as to have elements of comparable performance. Other strategies could have been considered and different results provided, but the purpose was only to show the use of controlled experiment for test strategy comparison and not to select the best one.

For future work, we plan to generalize our controlled experiment by considering different subjects, for instance the conformance test suite, different objects, i.e., different PDPs, and extend the set of considered dependent variables. Moreover, we plan to define and perform further controlled experiments in the context of access control for evaluating other testing techniques, for mutation generation or test suites reduction or prioritization.

ACKNOWLEDGEMENTS

This work is partially supported by CyberSec4Europe Grant agreement ID: 830929.

REFERENCES

(2019). Fedora Commons Repository Software. http://fedora-commons.org/.

Basili, V. R. and Rombach, H. D. (1988). The tame project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773.

Bertolino, A., Daoudagh, S., Kateb, D. E., Henard, C., Traon, Y. L., Lonetti, F., Marchetti, E., Mouelhi, T., and Papadakis, M. (2015). Similarity testing for access control. *Information and Software Technology*, 58:355 – 372.

Bertolino, A., Daoudagh, S., Lonetti, F., and Marchetti, E. (2012a). Automatic XACML Requests Generation for Policy Testing. In *Proc. of ICST*, pages 842–849.

Bertolino, A., Daoudagh, S., Lonetti, F., and Marchetti, E. (2012b). Modelling and Testing of XACML policies. In *2012-TR-010*.

Bertolino, A., Daoudagh, S., Lonetti, F., and Marchetti., E. (2013a). XACMUT: XACML 2.0 Mutants Generator. In *Proc. of 8th International Workshop on Mutation Analysis*, pages 28–33.

Bertolino, A., Daoudagh, S., Lonetti, F., and Marchetti, E. (2018). An automated model-based test oracle for access control systems. In *Proceedings of the 13th International Workshop on Automation of Software Test, AST@ICSE 2018, Gothenburg, Sweden, May 28-29, 2018*, pages 2–8.

Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., Martinelli, F., and Mori, P. (2014). Testing of polpa-based usage control systems. *Software Quality Journal*, 22(2):241–271.

Bertolino, A., Daoudagh, S., Lonetti, F., Marchetti, E., and Schilders, L. (2013b). Automated testing of extensible access control markup language-based access control systems. *IET Software*, 7(4):203–212.

Briand, L. and Labiche, Y. (2004). Empirical studies of software testing techniques: Challenges, practical strategies, and future research. *SIGSOFT Softw. Eng. Notes*, 29(5):1–3.

Daoudagh, S., Lonetti, F., and Marchetti, E. (2019a). A framework for the validation of access control systems. In Saracino, A. and Mori, P., editors, *Proceedings of the 2nd International Workshop on Emerging Technologies for Authorization and Authentication*.

Daoudagh, S., Lonetti, F., and Marchetti, E. (2019b). XACMET: XACML Testing & Modeling. *Software Quality Journal*.

Do, H., Elbaum, S., and Rothermel, G. (2004). Infrastructure support for controlled experimentation with software testing and regression testing techniques. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 60–70. IEEE.

Do, H., Elbaum, S., and Rothermel, G. (2005). Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering*, 10(4):405–435.

Elbaum, S., Malishevsky, A. G., and Rothermel, G. (2002). Test case prioritization: A family of empirical studies. *IEEE Transactions on Software Engineering*, 28(2):159–182.

Hu, C. T., Ferraiolo, D. F., Kuhn, D. R., Schnitzer, A., Sandlin, K., Miller, R., and Scarfone, K. (2019). Guide to attribute based access control (abac) definition and considerations [includes updates as of 02-25-2019]. Technical report.

Jedlitschka, A. and Pfahl, D. (2005). Reporting guidelines for controlled experiments in software engineering. In *Empirical Software Engineering, 2005. 2005 International Symposium on*, pages 10–pp. IEEE.

Jin, X., Krishnan, R., and Sandhu, R. (2012). A unified attribute-based access control model covering dac, mac and rbac. In *Data and Applications Security and Privacy XXVI*, pages 41–55, Berlin, Heidelberg. Springer Berlin Heidelberg.

Just, R. (2014). The major mutation framework: Efficient and scalable mutation analysis for java. In *Proceedings of the 2014 international symposium on software testing and analysis*, pages 433–436. ACM.

Juzgado, N. J. and Moreno, A. M. (2001). *Basics of software engineering experimentation*. Kluwer.

Li, Y., Li, Y., Wang, L., and Chen, G. (2014). Automatic xacml requests generation for testing access control policies. In *SEKE*, pages 217–222.

Ma, Y.-S., Offutt, J., and Kwon, Y.-R. (2006). MuJava: a mutation system for Java. In *Proceedings of the ICSE*, pages 827–830.

Madeyski, L. and Radyk, N. (2010). Judy - a mutation testing tool for java. *IET Software*, 4(1):32–42.

Martin, E. and Xie, T. (2006). Automated test generation for access control policies. In *Supplemental Proc. of ISSRE*.

Martin, E. and Xie, T. (2007). Automated test generation for access control policies via change-impact analysis. In *Proc. of SESS*, pages 5–11.

Martin, E., Xie, T., and Yu, T. (2006). Defining and measuring policy coverage in testing access control policies. In *Proc. of ICICS*, pages 139–158.

Mouelhi, T., Fleurey, F., and Baudry, B. (2008). A generic metamodel for security policies mutation. In *Proc. of ICSTW*, pages 278–286.

OASIS (22 Jan 2013). eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

Pretschner, A., Mouelhi, T., and Le Traon, Y. (2008). Model-based tests for access control policies. In *Proc. of ICST*, pages 338–347.

Runeson, P. and Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131.

Sandhu, R. S. and Samarati, P. (1994). Access control: principle and practice. *IEEE Communications Magazine*, 32(9):40–48.

Schuler, D. and Zeller, A. (2009). Javalanche: Efficient mutation testing for java. In *Proceedings of ESEC/FSE*, pages 297–298, New York, NY, USA. ACM.

seung Ma, Y., Offutt, J., and Kwon, Y. R. (2005). Mujava : An automated class mutation system. *Journal of Software Testing, Verification and Reliability*, 15:97–133.

Sun Microsystems (2006). Sun's XACML Implementation. http://sunxacml.sourceforge.net/.

TAS3 Project. Trusted Architecture for Securely Shared Services. https://cordis.europa.eu/project/rcn/85331/results/en.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.