Cluster Computing (2023) 26:3957–3972
https://doi.org/10.1007/s10586-022-03778-x

XACSim: a new tool for measuring similarity of XACML security policies

Zahra Katebi[1] · Mohammad Nassiri[1] · Mohsen Rezvani[2]

Received: 14 January 2022 / Revised: 18 September 2022 / Accepted: 3 October 2022 / Published online: 8 November 2022

Abstract
XACML is a standard to define a declarative fine-grained, attribute-based access control security policy language. Evaluation of the similarity of XACML policies can be used for a variety of purposes such as clustering rules, merging policies, analysing anomalies in rules, selecting high-speed web servers, and finding collaborators with similar security settings. Existing approaches for calculating the similarity between security policies are primarily designed based on the XACML 2.0 version, and are insufficient for complicated policies can be specified in XACML 3.0. In this paper, we propose a hierarchical approach, called XACSim, to assess the similarity of security policies specified by XACML 3.0. XACSim takes into account the distance of both numerical and nominal values for computing the similarity. More precisely, the distance is hierarchically computed by the aggregate of the distance values at four different levels namely, value, attribute, rule, and policy. For nominal attributes, the similarity is calculated based on their context and using distribution of their values in the input dataset. While, for numerical attributes, intersection intervals of their corresponding values are estimated to compute the similarity. We present an empirical evaluation of the effectiveness and efficiency of XACSim. The evaluation results show that our approach provides promising efficiency while it outperforms the effectiveness of the state of the art methods. (The XACSim tools are publicly available at https://gitlab.com/nassirim/XACSim.)

Keywords Access control · XACML · Similarity degree · Attribute distribution · Context · Policy Comparison

1 Introduction

The provision of data security is a fundamental requirement of any information system and access control is a crucial mechanism in data security. Using access control, administrators, and resource owners can protect their critical resources against unauthorized access and manipulation. After logging in to a system, the access control module decides about the user's access to specific resources based on some security policies defined by the system administrator. Extensible Access Control Markup Language (XACML) is an attribute-based language to describe access control security policies in web applications and is represented by XML language [1–4].

Access control facilitates the sharing and protection of resources for different organizations. Therefore, organizations can easily share their resources for secure collaboration in cloud computing applications. In these environments, however, each organization needs to know whether the access control policies used to share the resources are still required after sharing the resources. Thus, the comparison between access control policies and measurement of their similarities are an urgent need. As a result, a measure should be defined to specify the degree of similarity of every two access control policies. This measure can be used to merge policies with shared security features and share resources among several organizations [5].

Evaluation of the similarity of policies can be used for a variety of purposes such as clustering rules, merging policies, analyzing anomalies in rules, selecting high-speed web servers, finding collaborators with similar security settings [3, 6–12], and mutation analysis for different types

✉ Mohammad Nassiri
m.nassiri@basu.ac.ir

Zahra Katebi
z.katebi@alumni.basu.ac.ir

Mohsen Rezvani
mrezvani@shahroodut.ac.ir

[1] Computer Department, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran

[2] Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

of policies [13, 14]. The existing solutions to measure the similarity of XACML policies are mainly based on the older version of the standard, i.e., XACML 2.0 [6, 7, 15–17]. Most of these approaches tried to construct tree-based structures to examine the relationship between the values in each component of the XACML policies, such as `resource` and `action`. Establishment of such structures for calculating the similarity degree, however, is particularly difficult in XACML 3.0 given the broad variety of attributes as well as its hierarchical and nested nature of policies. In fact, the XACML 3.0 standard provides a more complex structure for policy specification which makes the similarity assessment more complicated. For example, the newer version of XACML proposes a dynamic and more complex syntax for defining the `target` elements along with additionally combining algorithms for resolving conflicts [1]. Therefore, a comprehensive similarity assessment method for XACML must be both simple and able to find relations among the values of the various attributes in the policies with the aim of calculating the degree of similarity in the case of any semantic relationship.

This paper proposes a novel and distance-based approach, called *XACSim*, to measure the similarity of security policies specified using XACML 3.0. The hierarchical structure of XACML 3.0 policies entails that similarity is calculated at all levels and among all components. Therefore, XACSim assesses the similarity of two policies at four levels, i.e., value, attribute, rule, and policy. The similarity of two policies is, then, an aggregate of the similarity degree at all four levels. Moreover, XACSim calculates the similarity of both categorical and numerical values using two different mechanisms. It computes the similarity of categorical values using the Distance Learning for Categorical Attribute (DILCA) [18] method, while it employs the concepts of distance and range intersection for numerical values. In addition to exact matching, our similarity measure can recognize related policies and estimate their similarity. DILCA calculates the similarity between categorical values based on the distribution of attributes in policies. For example, the attribute *city* may contain the following values: Rome, Paris, Florence. Obviously, Rome is more similar to Paris in that they are both capital cities. Geographically, however, Florence and Rome are more similar due to their shorter distance, which can only be recognized by humans [18]. Through application of our suggested mechanisms to the XACML Standard, a similarity evaluator can be developed.

We implement a prototype of XACSim as a software tool in the Java environment. We believe the XACSim tool can help security administrators to take into account different viewpoints that only humans can recognize. Also, using a bottom-up mechanism in XACSim that is based on the hierarchical structure and the semantics of XACML 3.0 policies, the proposed approach calculates the similarity of the components of policies and, ultimately, of two policies. We also conduct extensive experiments using both real-world and synthetically generated XACML policies to evaluate the efficiency and effectiveness of XACSim. The evaluation results show that the proposed approach can effectively assess the similarity of XACML policies using an acceptable amount of computation resources.

In summary, we make the following contributions:

- We propose a measure to calculate the similarity of categorical values according to the context and the distribution of values. This measure helps in identifying two policies with identical contexts and similar meanings.
- Concerning numerical values, the similarity is computed through the distance. We normalize the distance of values so that the difference in the scales does not affect the numerical similarity.
- In the proposed method, values along with their corresponding functions are extracted hierarchically from relevant rules.
- We also propose a hierarchical algorithm to calculate the similarity of two XACML 3.0 policies.
- We develop a software tool to measure the similarity of security policies in XACML 3.0.
- We conduct extensive experiments to evaluate the performance of our proposed approach and compare it with the state of the art.

The rest of the paper is organized as follows: Sect. 2 briefly describes XACML 3.0. Section 3 presents the related work. Section 4 introduces the proposed method. Next, Sect. 5 discusses the implementation and experimental results. Finally, Sect. 6 concludes the paper and suggests a direction of further research.

2 XACML Standard

The XACML Standard provides an access control policy language based on attributes and XML. So far, three versions of this standard have been released. The main components of a policy defined in XACML are `policy set`, `policy`, `rule`, `target`, `environment condition`, and `combining algorithm`. This request/response standard has a particular architecture and a processing model that defines how requests should be evaluated according to `policy sets`, `policies`, and `rules` defined in the policy [19–23].

The root of every XACML document contains either a policy or a policy set (see Fig. 1). Every policy set contains a target, a combining algorithm, and one or more policies. Every policy contains a target, a combining algorithm, and

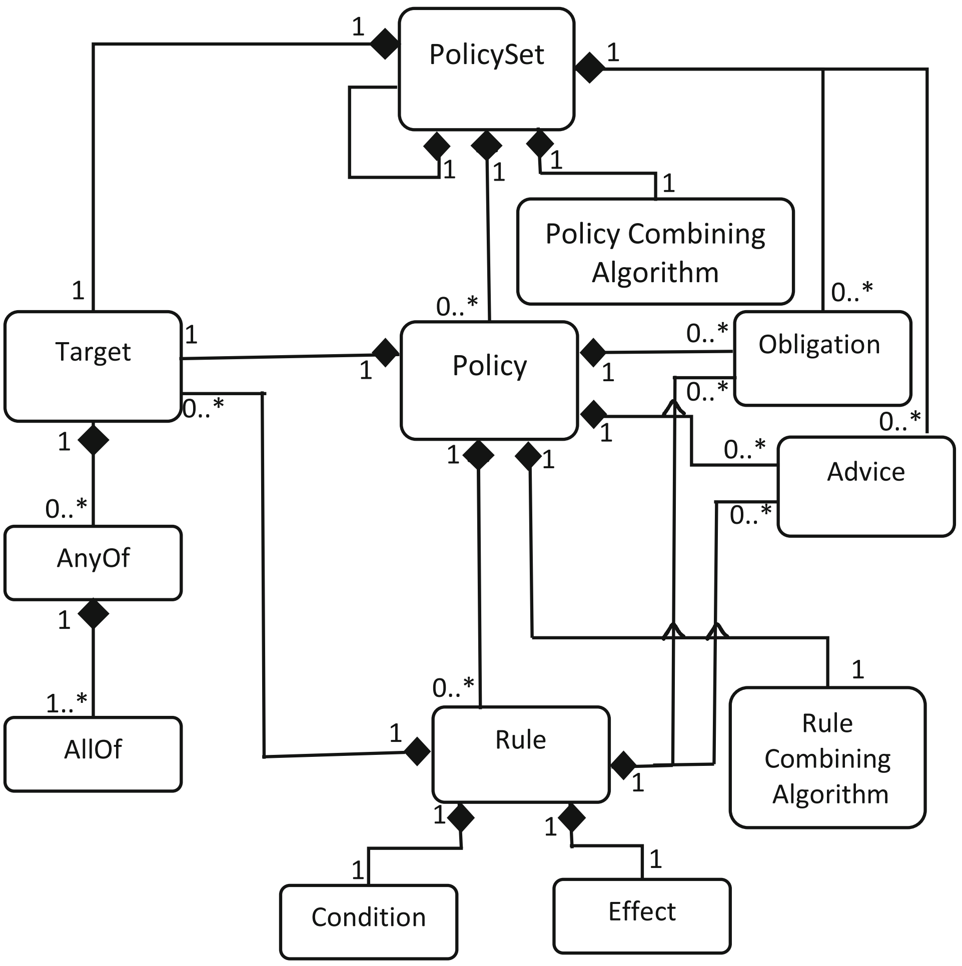


Fig. 1 XACML policy language model v3.0 [1]

one or more rules. In addition to a target, every rule includes an effect, a condition and may also contain obligation and advice. The effect of a rule indicates the consequence of evaluating the rule which can be either Permit, Deny, or Indeterminate. The target of a rule defines the applicability of the rule to a set of access requests. A target may contain a conjunctive sequence of zero, one, or more AnyOf components. An AnyOf component contains a disjunctive sequence of one or more AllOf components. An AllOf component contains a conjunctive sequence of one or more Match elements against which the user's request is matched. A Match element contains an Attribute-Value as well as an Attribute-Designator or an Attribute-Selector.

There are four popular combining algorithms in XACML including Permit-Override, Deny-Override, First-Applicable, and Only-One-Applicable. The user's request may match several rules in a policy or several policies in a policy set, but because the evaluation engine, called policy decision point (PDP), should return a single access level to the user, the combining algorithms can help to return a single result [1, 24, 25].

3 Related work

Analysing XACML policies with regard to the verification of policy properties, policy refinement and reconfiguration have already received the attention of research community [26–28]. Calculating similarity between two XACML policies is also necessary for several applications. By using a similarity measure, the number of policies to be evaluated can be reduced since similar policies could provide the same decisions. In [11, 15], the authors defined similarity to allow comparison of access control policies to locate providers that have similar policies in cloud environments.

Vaidya et al. [5] proposed the *XyDiff change detection tool* to measure the similarity of two XACML 2.0 policies. They also used their measure to address the problem of policy migration in collaborative environments at the lowest transition cost. Lin et al. [15] developed a method for calculating similarity of two XACML 2.0 policies considering both nominal and numerical attributes. They also integrated dictionary lookup and ontology matching. Their measure can be used as a filter phase to quickly reduce the number of policies for further analysis [29].

Although we take a similar hierarchical approach for numerical attributes, we propose a context-based measure for nominal attributes.

Yan Hung et al. [6] proposed a similarity measure between two security policies and categorized every policy into two rule sets. They separately calculated similarity in each set in a hierarchical manner. For categorical attributes, they measure the exact match of two values. In other words, a higher score indicates that the two attributes share more common attribute values. El-hadej et al. [7] used similarity to reduce redundancy among XACML 2.0 rules by either eliminating or merging them into other rules. Lin-Lee et al. [30] proposed to calculate the weight of each attribute in all XACML 2.0 rules. Next, they calculated the similarity between policies for categorical and numerical values in a hierarchical manner. Also in [31], Bretia et al. developed a hierarchical similarity analyzer (HSA) for policies that should be merged. They calculated the key component SL (Security Level) and allocated it to users who share their data. SL refers to the amount of data that is allowed to be shared during the collaboration of two policies. This measure can be used to find collaborators with similar security settings. The problem of policy similarity have been recently investigated for ABAC policies [29, 32, 33]. Batra et al. in [29] proposes a policy similarity metric for two ABAC policies based on the maximum common possible subset of accesses covered by their rules.

El Hadj et al. [27, 34] are attempts to identify redundancy and interference between rules in a policy. They calculate the similarity in order to cluster the rules and eliminate redundancy. This approach detects and eliminates redundancy by clustering the rules and calculating the similarity of the rules in each cluster. On the other hand, Lenco et al. [18] investigated the relation between attribute values by calculating the distance which was then used to cluster attributes. However, their work did not rely on XACML. We employ a similar approach to measure the similarity between values of the nominal attributes without establishing a relation between the resource and the subject for the attributes.

As mentioned earlier, the literature has mostly focused on older versions of XACML, and the proposed similarity measures are almost impractical in the context of the variety of attributes in XACML 3.0.

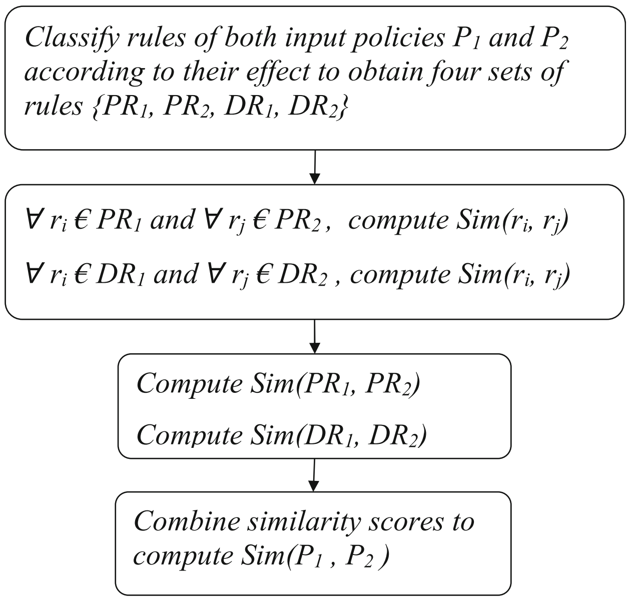
4 Context-based similarity measurement

In this paper, we propose a context-aware similarity assessment between two policies specified in XACML, we call it XACSim. In this section, we first describe the conceptual framework of XACSim including an overview of the main steps of the similarity assessment. We then explain the details of its components.

4.1 Solution overview

Figure 2 summarizes the main steps in XACSim for assessing the similarity of two policies specifies in XACML. As shown in the figure, the similarity between two policies is calculated hierarchically according to the hierarchical structure of the XACML standard, as explained in Sect. 2. First, every policy is divided into two rule sets, namely, Permit (including the rules with a permit effect) and Deny (containing the rules with a deny effect). Next, the similarity of each pair of rules in both input policies is computed by comparing their respective attributes. The similarity of the attributes is calculated by comparing their values. By finding the average degree of similarity between two rule sets, the similarity of Permit and Deny sets is calculated. Finally, the degrees of similarity of the components of the two policies are added to obtain the degree of similarity at a policy level. Details about each step is discussed in Sects. 4.2 to 4.5. In general, XACSim assigns a higher degree of similarity to policies with more similar values for attribute types. It should be noted that the total degree of similarity is a value in the range [0, 1].

Two policies are called equal if they possess completely identical attribute values, i.e., their similarity is 1. On the other hand, two policies are unequal if they have at least one unequal attribute value. Unequal policies are divided into "partially similar" and "completely dissimilar". By way of example, we consider two unequal policies that are partially similar in terms of their meaning. The first policy



Classify rules of both input policies P_1 and P_2 according to their effect to obtain four sets of rules $\{PR_1, PR_2, DR_1, DR_2\}$
$\forall r_i \in PR_1$ and $\forall r_j \in PR_2$, compute $Sim(r_i, r_j)$
$\forall r_i \in DR_1$ and $\forall r_j \in DR_2$, compute $Sim(r_i, r_j)$
Compute $Sim(PR_1, PR_2)$
Compute $Sim(DR_1, DR_2)$
Combine similarity scores to compute $Sim(P_1, P_2)$

Fig. 2 A high-level view of the main computation steps in XACSim

(P_1) specifies people in Rome who have only access to POP3 email service. In contrast, the second policy (P_2) specifies females living in Florence who have access to both SMTP and POP3 services. The attribute types of these two policies include "population", "city", and "email service" with their values consisting of "females", "Rome", "Florence", "POP3", and "SMTP".

In this example, we examine how the similarity of Rome and Florence can be discovered. As the two values, Rome and Florence are different, their relation can be found employing the similarity of other values within this policy. Should two policies have at least one common value that is related to Rome and Florence, then the obtained similarity of Rome and Florence is greater than 0; otherwise, it becomes 0.

Here, the point of similarity between the two policies is the group of females who have access to POP3 service. As a result, the obtained similarity is greater than 0 due to a shared value that is related to both Rome and Florence. Following the same procedure, we can hierarchically calculate the similarity between attributes (categorical and numerical) and rules, in which we present the detailed computation in the rest of this section.

Our proposed solution calculates similarity by comparing the components of XACML policies, which is generally similar to the method proposed in [15]. Components here refer to the target elements of the rules within policies. This comparison of components is performed on rules with similar effects. We categorize the rules of policies P_1 and P_2 according to their effects, resulting in two Permit rule sets PR_1 and PR_2 and two Deny rule sets DR_1 and DR_2, as shown in Fig. 2. Each rule in P_1 is compared with all rules in P_2 having a similar effect to calculate the degree of similarity of those two rules. Thus, the degree of similarity between all rules as well as between both Permit rule sets (PR_1 and PR_2) and between both Deny rule sets (DR_1 and DR_2) are calculated. To calculate the similarity between two rule sets with similar effects, we define a *mapping functions* denoted by map. The map functions relate two rules with the highest degree of similarity to each other. Thus, we first explain the details of the mapping calculation, and then the computation details of each element are separately described.

4.2 Mapping computation and policy similarity

Since we are employing a hierarchical similarity assessment between XACML security policies, an important concept is to compute the similarity of a rule and a rule set. Moreover, we need to generalize such concepts to calculate the similarity of two rule sets, for example, between two permit rule sets extracted from two input policies. Thus, we define the concept of the *map* as the similarity degree of a rule and a rule set as follows:

Definition 1 (**Rule Map**) The map of rule r to rule set P, denoted as $\mathrm{map}(r,P)$, is a measure of the similarity of rule r to rule set P, defines as

$$\mathrm{map}(r,P) = \max_{r_i \in P}\{\mathrm{sim}(r,r_i)\} \tag{1}$$

where $\mathrm{sim}(r,r_i)$ denotes the similarity score of two rules r and r_i.

As one can see in Definition 1, we employ a *single-link* approach, where the similarity of a rule and a rule set is equal to the similarity of that rule with the most similar member in the rule list. This single-link criterion is local. In other words, we pay attention solely to the area where that rule comes closest to the rule set. It is to be noted that we can also use a *complete-link* approach, where the most dissimilar link is considered for such computation. Such non-local criterion results in a preference for a compact rule set with small diameters over long, straggly sets, but also causes sensitivity to outliers. Moreover, our experiments show an improvement using the single-link criterion. More specificity, since we only check the similarity of rules sets with similar effect values, a local criterion is a suitable alternative.

Definition 2 (**Rule set Map**) The map of rule set P_1 to rule set P_2, denoted as $\mathrm{map}(P_1,P_2)$, is a measure of the similarity of rule set P_1 to rule set P_2, defines as

$$\mathrm{map}(P_1,P_2) = \sum_{r_i \in P_1}\mathrm{map}(r_i,P_2) = \sum_{r_i \in P_1}\max_{r_j \in P_2}\{\mathrm{sim}(r_i,r_j)\} \tag{2}$$

where $\mathrm{map}(r_i,P_1)$ denotes the map of r_i to rule set P_1, and $\mathrm{sim}(r_i,r_j)$ denotes the similarity score of two rules r_i and r_j.

Proposition 1 *The rule set map is not a symmetric property. In other words, we have:*

$$\exists P_1,P_2 : \mathrm{map}(P_1,P_2) \neq \mathrm{map}(P_2,P_1) \tag{3}$$

Proof In order to prove this proposition, we only need to provide an example holding such inequality. To this end, it is enough to consider two rule sets with a different number of members. For example, assume that $N_{P_1} = 1$ and $N_{P_1} = 10$. In this case, we can clearly see that it is highly possible that $\mathrm{map}(P_1,P_2) \ll \mathrm{map}(P_2,P_1)$. □

As mentioned in the previous section, XACSim first divides every input policy into two rule sets, permit and deny rule sets. Consequently, it generates four rule sets from two input policies. Next, we employ the map formulas to separately compute the similarity of rule sets with the same effect value; for example, the similarity of permit rule sets and similarity of deny rule sets.

Since the map of two rule sets is not symmetric, we need to consider the map on both sides, and then normalize the obtained values. We repeat this procedure for both permit and deny rule lists obtained from two input policies. Thus, we compute the similarity of permit and deny pairs, as follow:

$$\begin{cases} S^P(P_1,P_2) = \dfrac{\text{map}(PR_1,PR_2)+\text{map}(PR_2,PR_1)}{|PR_1|+|PR_2|} \\ S^D(P_1,P_2) = \dfrac{\text{map}(DR_1,DR_2)+\text{map}(DR_2,DR_1)}{|DR_1|+|DR_2|} \end{cases} \quad (4)$$

where P_1 and P_2 are two input XACML policies. PR_1 and PR_2 (DR_1 and DR_2) are the permit (deny) rule sets of P_1 and P_2, respectively. For a set S, $|S|$ indicates the number of elements in the set. $S^P(P_1,P_2)$ is the similarity score of permit rule sets obtained from policies P_1 and P_2. $S^D(P_1,P_2)$ is the similarity score of deny rule sets obtained from policies P_1 and P_2.

Now, we can compute the similarity of two XACML policies as the sum of the similarity scores of their components, including their Permit and Deny rule sets and their target elements, as follows:

$$\begin{aligned} S(P_1,P_2) =& w_t \times S_T(P_1,P_2) + w_p \times S^P(P_1,P_2) \\ &+ w_d \times S^D(P_1,P_2) \end{aligned} \quad (5)$$

where $S(P_1,P_2)$ indicates the similarity score of two XACML policies P_1 and P_2. Also, $S_T(P_1,P_2)$ denotes the degree of similarity of the target elements of these two policies. In equation (5), w_t, w_p and w_d are three constants in the range $0 \le w_t \le 1$, $0 \le w_p \le 1$, $0 \le w_d \le 1$ and $w_t + w_p + w_d = 1$, chosen to reflect the impact of each policy element in the computation of the similarity score.

4.3 Similarity of rules

The similarity of two rules is calculated hierarchically by comparing the corresponding components of the two rules. We consider such similarity only between two rules with a common effect value that we categorized them in a the same category in the previous step. Each rule has a target component containing a set of attributes. More precisely, we compute the similarity of two rules based on the similarity of their common attribute types. Moreover, we need to take the intrinsic difference of categorical and numerical attributes into the account in our computation. The similarity score of rules r_i and r_j is obtained as:

$$\text{sim}(r_i,r_j) = \begin{cases} 0 & \text{if } A_{r_i} \cap A_{r_j} = \emptyset \\ 1 & \text{if } |A_{r_i}| + |A_{r_j}| = 0 \\ \dfrac{\sum_{a \in A_{r_i} \cap A_{r_j}} S(r_i,r_j,a)}{|A_{r_i}| + |A_{r_j}|} & \text{if } A_{r_i} \cap A_{r_j} \neq \emptyset \end{cases} \quad (6)$$

where, $S(r_i,r_j,a)$ denotes the similarity of two rules r_i and r_j based on their common attribute type a. Also, A_{r_i} is the set of attribute types in rule r_i.

As one can see in equation (6), we compute the similarity of two rules based on the similarity of their common attribute types. It is obvious that the similarity of two rules with no common attribute type is 0 as their attributes are completely different. Likely, the similarity of two rules with no attributes in their target elements is 1 as their target elements are identical. Note that we compute the similarity only for two rules with common effect value.

We also employ two separate similarity functions for categorical and numerical attribute types. Both of these two functions are defined based on the attribute values presented in each rule. Thus, we need to extract a list of values of the comparing attribute for each rule. Then, we employ two similarity functions to find the similarity of two list of values, categorical and numerical values. Thus, we have:

$$S(r_i,r_j,a) = \begin{cases} S_{cat}(V^a_{r_i}, V^a_{r_j}) & \text{if } a \text{ is categorical} \\ S_{num}(V^a_{r_i}, V^a_{r_j}) & \text{if } a \text{ is numerical} \end{cases} \quad (7)$$

where $V^a_{r_i}$ is the set of all values of attribute a presented in rule r_i. Also, S_{cat} and S_{num} are two functions for computing the similarity of two set of values in categorical and numerical types, respectively.

4.4 Similarity of attributes

As mentioned in the previous section, the similarity score of two identical attribute types in two rules is obtained by the sum of similarities of their corresponding values and varies with the type of their values. More precisely, in order to compute the similarity of two rules based on a common attribute type within them, we first extract two sets of all possible values matched in each of these two rules. Then, we compute the similarity of these two sets of values using two similarity functions, one corresponding for the categorical and another for numerical values.

The similarity between two list of categorical values L_1 and L_2 is computed as follows:

$$S_{cat}(L_1,L_2) = \frac{\sum_{v \in L_1} S_{cat}(v, L_2)}{N_{L_1}} \quad (8)$$

$$S_{cat}(v,L) = \begin{cases} 1 & \text{if } \exists v' \in L : v = v' \\ \dfrac{\sum_{v' \in L} S_{cat}(v,v')}{N_L} & \text{if } \nexists v' \in L : v = v' \end{cases} \quad (9)$$

where L_1, L_2 and L are sets of categorical values from an identical attribute type, N_L indicates the number of elements in list L. We assume $N_{L_1} \ge N_{L_2}$. Also, v and v' are two categorical values from an identical attribute type. It is to be noted that we defined here three overloaded similarity functions that are different in their argument types. More clearly, we have a function with two lists of values, a function with a value and a list of values, and a function with two attribute values in the input arguments. We describe the detailed computation of the last function for computing the similarity of two categorical values in Sect. 4.5.1.

We employ a similar method for calculating the similarity of two lists of numerical range values, as referred in equation 7. Each element of this list represents a range of numerical values. Since XACML standard allows us to use comparison operators for numerical attributes, we represent the matched values of an attribute type in a target element using a list of range values. We describe the details of this algorithm in Sect. 4.6.

The similarity of two sets of range values of a single attribute type L_1 and L_2 is computed as follows:

$$S_{num}(L_1, L_2) = \frac{\sum_{v \in L_1} \max_{v' \in L_2} S_{num}(v, v') + \sum_{v \in L_2} \max_{v' \in L_1} S_{num}(v, v')}{N_{L_1} + N_{L_2}} \tag{10}$$

where $S_{num}(v, v')$ is the function for computing the similarity of two range values of v and v' obtained from the domain of a single attribute type.

As one can see, we employ a single-link technique for computing the similarity of two sets of numerical range values, while for the categorical values, we use an average-link method. This is because of the fact that we can define an intersection relation among numerical range values, so a single-link approach considers the contribution of the largest intersection among the range values for computing the similarity. However, it is tough to define a subset/superset relationship among the contextual values, so we consider the contribution of all values in the set for computing the similarity by using an average-link technique. It is to be noted that in average link and for two sets R and S, first for the distance between any data-point i in R and any data-point j in S and then the arithmetic mean of these distances are calculated. However, the single linkage returns the minimum distance between two points i and j such that i belongs to R and j belongs to S [35].

4.5 Similarity of attribute values

In order to compute the similarity of two values, we consider the following two states: 1) if the values exactly match each other, the degree of similarity is 1; If the values are not equal, we compute the similarity of the numerical values using the length of the intersection of the range values, while for the categorical values, the similarity is calculated similar to the DILCA method [18]. In the subsequent subsection, we explain how to calculate the similarity for both types of values.

4.5.1 Similarity of categorical values

To calculate the similarity of two unequal categorical values, the distance between the values is calculated similar to the DILCA method proposed in [18]. Here, we provide several definitions to explain this approach:

- *Target* attribute: Both attributes to be compared are called target attributes.
- *Context* attributes: Attributes that have frequently co-occurred with the values of the target attributes in the input policy.
- *Entropy* of attribute X: The distribution of the values of an attribute X in the input policy is called the entropy of that attribute.
- *Information Gain* of an attribute: The amount of information needed by an attribute to partition related attributes.
- *Heteronymous* attributes: Attributes with identical types but different names and values.
- *Homonymous* attributes: Attributes with identical names and types but not necessarily identical values.
- *Conditional entropy* $(X \mid Y)$: The distribution of the values of an attribute X in cases where they co-occur with the values of an attribute Y in the input policy is called the conditional entropy of X given that Y.

The DILCA method is a context-based approach consists of three steps: 1) finding the frequency of values, 2) selecting context attributes for every target attribute, and 3) calculating the distance using context attributes. We now briefly explain the details of these three steps.

Finding the frequency of values: the first step is to find the frequency of values in the input XACML policy. More precisely, we use the distribution of values within policies which refers to the frequency of co-occurrence of each pair of categorical values in the policies. Thus, the frequency of co-occurrence of values within one rule of a policy should be calculated. This is done for all rules and policies in the input data. For example, suppose that a given policy has only one policy element with a single rule, and the rule has only two attributes, namely, *book author* (X) and *book subject* (Y). The values of *author* and *subject* are respectively $\{x_1, x_2, x_3\}$ and $\{y_1, y_2\}$ in the input policy. If the data distribution pattern resembles what is shown in Table 1, the frequency of co-occurrence of values is represented by the distribution shown in Table 2. Each item in the table represents the frequency of the co-occurrence of the values in its row and columns. A distribution table should be provided for each target attribute and between every two heteronymous attributes in each rule (e.g. X and Y).

Selection of context attributes: To specify the context of attributes, we need to select a subset of relevant and non-overlapping attributes. To this end, we use Symmetric Uncertainty approach which has already been used in [18] to measure the correlation between two variables. Here, we recall how Symmetric Uncertainty is derived from entropy according to the information theory. Using the frequency distribution table of two given attributes (e.g. X and Y), the entropy of attribute X is computed as follows:

$$H(X) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i) \tag{11}$$

where $p(x_i)$ is the probability of the value x_i of the attribute X. After having observed the values of attribute Y, the conditional entropy of X is calculated as [8, 23, 36]:

$$H(X|Y) = -\sum_{x_j \in Y} p(y_j) \sum_{x_i \in X} p(x_i|y_j) \times \log_2 p(x_i|y_j) \tag{12}$$

where $p(x_i \mid y_j)$ denotes the probability that $X = x_i$ after observing y_j and $p(y_j)$ is the frequency of y_j in the input policy. The context attributes should be calculated separately for the two target attributes in the policies P_1 and P_2. For the target attribute X_1 (X_2) that belongs to the policy P_1 (P_2), the context attributes are selected from the attributes of the policy P_1 (P_2).

The information about attribute X provided by Y is given by the information gain defined as follows:

$$IG(X|Y) = H(X) - H(X|Y) \tag{13}$$

For two attributes X and Z, $IG(X|Y)$ means that X is more correlated to Y than Z [8].

Equation (13) represents information Gain. According to this measure, the attribute Y is more dependent on the attribute X than the attribute Z if $IG(X|Y) > IG(Z|Y)$. Information Gain for two random variables X and Y is symmetrical, which means that the information Gain of Y after observing X has become equal to the information Gain of X after observing Y [8]. Information Gain has become biased on attributes with a larger number of values. That is, attributes with a larger number of values have greater information Gain than attributes with a smaller number of values. In addition, the values should be normalized to ensure the accuracy and validity of comparisons. Therefore, there is a need to define and use a measure called symmetrical uncertainty. Symmetrical uncertainty (SU): To compensate for the bias of information Gain on attributes with larger values, this measure normalizes the values in the range [0,1] in which 1 indicates that each value perfectly predicts the other variable whereas 0 indicates total independence between X and Y. Thus, a pair of attributes behave symmetrically [8]. Equation (14) calculates symmetrical uncertainty.

Table 1 Sample input dataset

X	Y
x_1	y_1
x_2	y_2
x_1	y_1
x_3	y_2
x_2	y_1

Table 2 Contingency table for the dataset shown in Table 1

X/Y	y_1	y_2
x_1	2	0
x_2	1	1
x_3	0	1

$$SU(X, Y) = 2 \times [\frac{IG(X|Y)}{H(X) + H(Y)}] \tag{14}$$

Now, we define two attributes as context attributes if their symmetric uncertainty value is greater than a threshold value. Thus, we define the set of context attributes for an attribute X as:

$$Context(X) = \{Y \mid Y \neq X,\ SU(X, Y) \geq SU_{threshold}\} \tag{15}$$

where X and Y are the target and context attribute, respectively [18]. Also, $SU_{threshold}$ is the threshold value. A higher threshold value results in a fewer context attributes leaving room for only those attributes that have higher distribution with the target attribute. In this paper, the threshold value ($SU_{threshold}$) was set to 0.5 as a result of a trial and error mechanism.

Calculating the distance using context attributes: To calculate the distance between two values, the context attributes obtained in the previous step are used. Here, only common context attributes belonging to both target attributes will affect the distance. Equation (16) calculates the distance between two categorical values v and v'. Recall that v and v' are two values of the target attribute X in policies P_1 and P_2 respectively. Also, Y is a context attribute of X. Eqs. (17) and (18) define the values $P(v \mid y_k)$ and $P(v' \mid y_k)$. $P(v \cap y_k)$ denotes the co-occurrence probability of v and y_k and $P(y_k)$ is the probability of occurrence of y_k in the entire input policy where y_k is a value of the attribute Y.

$$D_{cat}(v, v') = \sqrt{\sum_{Y \in Context(X)} \sum_{y_k \in Y} ((P(v|y_k) - P(v'|y_k))^2} \tag{16}$$

$$P(v|y_k) = \frac{P(v \cap y_k)}{P(y_k)} \quad (17)$$

$$P(v'|y_k) = \frac{P(v' \cap y_k)}{P(y_k)} \quad (18)$$

Now, the similarity between two categorical values is the inverse of the distance [37] and computed as follows:

$$S_{cat}(v, v') = \frac{1}{D_{cat}(v, v') + 1} \quad (19)$$

where $S_{cat}(v, v')$ is always in the range of 0 and 1.

4.5.2 Calculation of similarity between numerical values

The similarity of two numerical ranges values is a normalized score computed based on the distance between the ranges as follows:

$$S_{num}(v, v') = e^{-D_{num}(v,v')} \quad (20)$$

where v and v represent the numerical ranges values in policies P_1 and P_2, respectively. In addition, $e \simeq 2.7$ is a constant value, and $D_{num}(v, v')$ denotes the distance between two numerical ranges values v and v'. Clearly, we assume that all numerical values are in the form of ranges. Therefore, the intersection between two numerical ranges can be used in calculating their distance. The intersection of two range variables v and v', denoted by $(v \cap v')$, is a new numerical range value that we represent as $[m, n]$, where m and n are the minimum and maximum of the range, respectively.

Moreover, we need to transform the range value into a numerical value, which helps for computations as well as working with attributes in different scales. Here we use the following transformation:

$$g([m,n]) = \frac{|n-m|}{|n+m| - |n-m|} = \begin{cases} \frac{n-m}{2m} & \text{if } m < n \\ 0 & \text{Otherwise} \end{cases} \quad (21)$$

The distance is, now, computed as:

$$D_{num}(v, v') = \frac{\max(g(v), g(v'))}{g(v \cap v')} \quad (22)$$

4.6 Extraction of the values of attributes from a rule

As mentioned above, we need the list of all values of attributes in the input policy for our similarity computation. Each rule contains a set of attributes with categorical and numerical values. These values are extracted hierarchically from the target element of the rule. The extraction of these values begins from the smallest member of the target, i.e. the match component. As every match component has a value in its *Attribute_value* tag and a function in its *function_match* tag, the similarity of the related function should be taken into account in addition to that of the values while calculating the degree of similarity. The function refers to five operators, i.e. greater than, less than, less than or equal to, greater than or equal to, equal to. The extraction of values related to the attributes from a rule may address either categorical values or numerical values. In this method, all the values related to an attribute are extracted from a rule independently and separately from other attributes. We describe the extraction of both categorical and numerical values in the rest of this section.

4.6.1 Extraction of categorical values

With this type of value, each operator along with its value in a match component is converted into a range and hierarchically extracted from the corresponding rule. These ranges are called categorical ranges. For example, a match component holds the value L: "*computer*" in its *attribute_value* tag. The related function holds the operator "greater than" (>). According to the proposed method, the value and operator are extracted from such a match component in the form of L' : ("*computer*", >). For the extraction of categorical values from a rule, we need to define the intersection and union between the ranges.

Definition 3 (Intersection of Categorical Ranges) The intersection of categorical ranges refers to finding similar ranges among the existing ranges.

Definition 4 (**Union of Categorical Ranges**) The collection of categorical ranges into a list is called the union of ranges.

Every AllOf component consists of one or more match components. Equation (23) is used to find the intersection between the values extracted from every match component. This intersection results in none or only one range, which is related to an AllOf. In this equation, every L'_i $(1 \le i \le k)$ denotes the range created from a match component. Every AnyOf component consists of one or more AllOf components. After extracting the list of the ranges related to the AllOf components of an AnyOf, equation (24) can be used to create the union of the obtained lists of values (including categorical ranges). In equation (24), every n_i $(1 \le i \le k)$ represents a list of ranges that are related to an AllOf. Finally, a target consists of one or more AnyOf components. The intersection of the values in the extracted list of every AnyOf component is calculated using equation (25). In this equation, every t_i $(1 \le i \le k)$ denotes the list of ranges extracted from an AnyOf component. Thus, the target of a rule may result in a value or a list of values.

$$M_{allof} = [L'_1 \cap L'_2 \cap \ldots \cap L'_k] \tag{23}$$

$$N_{anyof} = [n_1 \cup n_2 \cup \ldots \cup n_k] \tag{24}$$

$$T_{target} = [t_1 \cap t_2 \cap \ldots \cap t_k] \tag{25}$$

4.6.2 Extraction of numerical values

We convert every numerical value obtained from a match component into a range according to its function type. The ranges should be extracted hierarchically from a target. For example, a match component has a value of 21 and a function of "less than" (<). It can be converted into the range $(21, \infty)$. Algorithm 1 is proposed to compute the intersection/union of two lists of numerical range values. First, all the unique limit values of all the intervals in both L_1 and L_2 are stored in a temporary list (S) and sorted in ascending order (lines 1–8). Then, using values in S, a new set of all possible consecutive intervals is generated (lines 9–12). Finally, by comparing each interval with both L_1 and L_2, the intersection and union of these two lists are calculated (lines 13–21).

Algorithm 1 Compute Intersection/Union of two lists of intervals

```
Input: $L_1, L_2$        ▷ List of interval variables of policy $P_1$ and $P_2$, respectively
Output: Intersect $L_1, L_2$ (Union $L_1, L_2$)
1: $bList \leftarrow \emptyset$ ▷ $bList$ is used to maintain the set of all bound values in both $L_1$ and $L_2$
2: for each interval $(a,b) \in L_1$ do    ▷ $a$ and $b$ are bound values of $L_1$
3:     $bList \leftarrow bList \cup \{a,b\}$;
4: end for
5: for each interval $(a,b) \in L_2$ do
6:     $bList \leftarrow bList \cup \{a,b\}$;
7: end for
8: $S \leftarrow$ aSort($bList$)    ▷ The resulting list, $S$, is sorted in ascending order.
9: $IL \leftarrow UL \leftarrow IVL \leftarrow \emptyset$
10: for $i \leftarrow 1$ to $length(S) - 1$ do    ▷ $IVL$ list will contain all possible consecutive intervals
11:     $IVL \leftarrow IVL \cup \{(S_i, S_{i+1})\}$
12: end for
13: for each $(a,b) \in IVL$ do
14:     if isIntersected($(a,b), L_1$) AND isIntersected($(a,b), L_2$) then
15:         $IL \leftarrow IL \cup (a,b)$    ▷ $IL$ is the Intersection of $L_1$ and $L_2$
16:     end if
17:     if isIntersected($(a,b), L_1$) OR isIntersected($(a,b), L_2$) then
18:         $UL \leftarrow UL \cup (a,b)$    ▷ $UL$ is the Union of $L_1$ and $L_2$
19:     end if
20: end for
21: return $IL(UL)$
```

4.7 Distance at different levels

This section addresses the calculation of the distance between components at different levels. Just as we calculated similarity by the means of a bottom-up mechanism for XACML 3.0 policies and measured it hierarchically for the components of policy at the levels of value, attribute, rule, and policy, we can also calculate the distance at different levels for the components of a policy. In what follows, the calculation of the distance at different levels is described.

The distance between two categorical values is calculated by equation (16) and the distance between two numerical values is calculated by equation (22), as described in Sect. 4.6.

4.7.1 Distance of categorical and numerical attributes at the level of attribute

Equation (26) represents the distance of two categorical attributes a_i and a_j (belonging to P_1 and P_2, respectively) which is the average of the distances between the values of these attributes. Equation (27) denotes the distance of two numerical attributes b_i and b_j (belonging to P_1 and P_2, respectively) which is the average of the minimum distances between the values of these attributes.

$$Dis_{cat}(a_i, a_j) = \frac{\sum_{v \in a_i} \sum_{v' \in a_j} D_{cat}(v, v')}{N_{a_i} + N_{a_j}} \tag{26}$$

$$Dis_{num}(b_i, b_j) = \frac{\sum_{v \in b_i} \min_{v' \in b_j} D_{num}(v, v') + \sum_{v \in b_j} \min_{v' \in b_i} D_{num}(v, v')}{N_{b_i} + N_{b_j}} \tag{27}$$

where N_{a_i} (N_{a_j}) is the number of values of a_i (a_j) in $P_1(P_2)$.

4.7.2 Distance at the level of rule/ruleset/policy

We described the similarity computation between two rules with similar effect values in Sect. 4.3. We employ a very similar method for computing the distance between two rules, avoid repeating the equations here.

Equation (28) represents the distance between two policies, which is equal to the sum of distances of their targets, their permit and deny rule sets. So, we have:

$$Dis(P_1, P_2) = D_T(P_1, P_2) + D^P(P_1, P_2) + D^D(P_1, P_2) \tag{28}$$

where $D_T(P_1, P_2)$ denotes the distance of the target elements of the two policies. Also, $D^P(P_1, P_2)$ and $D^D(P_1, P_2)$ are the distances of permit and deny rule sets of the policies, respectively. It is to be noted that we employ a hierarchical approach for computing these distance scores, which is very similar to the one we applied for calculating the similarity score. Thus, we avoid repeating the equations to save space.

As we described in Sect. 4.2, we employed a single-link approach to defining the similarity assessment between two rule sets. We use a very similar approach for computing the distance between two ruleset, $D^P(P_1, P_2)$ and $D^D(P_1, P_2)$. The only difference here is that we define the map function using a complete-link approach, where the distance of a rule and a rule set is equal to the minimum distance between the rule and all rules in the ruleset. Most of the equations presented in Sect. 4.2 are used for computing the distance between two rulesets; thus, we avoid repeating them here.

5 Implementation and evaluation

In this section, we detail the implementation of the proposed approach, and evaluate the performance of our approach for assessing the similarity and distance of the XACML policies.

5.1 Prototype implementation

We have implemented the XACSim tool[1] in Java with JDK 1.8 as a similarity and distance evaluator for XACML version 3.0 policies. For implementing the XACML navigator to parse policies, we used the Java Architecture for XML Binding (JAXB) API.[2]

5.2 Evaluation environment

We conducted several experiments to evaluate the effectiveness and efficiency of the proposed method. All these experiments were conducted on a system with these specifications: a dual-core Core i5 CPU with a 4 MB of cache memory and a maximum frequency of 2.7 GHz, 4 GB DDR-3 RAM, and running the Windows operating system. We synthetically generated the following four datasets for running these experiments. Our datasets were generated using the *XACBench* toolset developed in our research laboratory[3] [38].

– `DS1_Xacml3`: This dataset consists only of categorical values and has a policy set defined in XACML 3.0. It contains 100 policies, each comprising three rules (two with Allow and one with Deny effect) which each in turn has at most 28 attributes with a maximum of three different values.
– `DS2_Xacml3`: Defined in XACML 3.0, this dataset consists of both categorical and numerical values. Its policy contains 100 policies, each comprising at most three rules (two with Allow and one with Deny effect) which each in turn has at most 28 categorical attributes with two different values. The maximum number of attributes with identical values in each rule is 2.
– `DS3_Xacml3`: This dataset consists of an only categorical value, and has a policy set defined in XACML 3.0. The policy set has 100 policies, each containing a maximum of one rule (Deny or Permit). Each rule has a maximum of three attributes. This dataset was used to compare the degrees of similarity obtained by our proposed method and the approach proposed in [15].
– `DS4_Xacml2`: This dataset is the equivalent version of `DS3_Xacml3` in XACML 2.0.

Table 3 lists the specifications of these policy sets. `DS1_Xacml3` and `DS2_Xacml3` have 25 subsets, each consisting of four policies with equal number of context attributes. In other words, we have 25 subsets of policies differing in terms of the number of their context attributes.

Our method can calculate the similarity between policies and determine a similarity score for each pair of policies. Also, it can measure the distance between the two policies. An important parameter in our evaluation is the number of context attributes that could affect both similarity and distance. The threshold parameter was also examined as a factor that affects the similarity between two policies. To evaluate the efficiency of our method, the elapsed time of the calculation of similarity and distance between the policies was also taken into account.

We also compared our results with those of *PSM* [15]. In addition, we use the similarity score S_{policy} for any two given policies which indicates the fraction of requests obtaining the same decisions namely, `Permit` or `Deny`. This measure is calculated as follows and referred as *SPDP*:

$$S_{Policy}(P_1, P_2) \approx \frac{S_{req}}{|Req|} \qquad (29)$$

where S_{req} represents the set of the requests with the same decisions from P_1 and P_2 and Req is the set of the requests applicable to either P_1 or P_2.

We implemented these approaches in Java to separately evaluate the policies of XACML 2.0 and 3.0. SPDP uses the number of user's requests and employs a request evaluator, called PDP, to calculate the similarity between two policies.

[1] The source code is publicly available at https://gitlab.com/nassirim/XACSim. The developed tool is executed as a JAR file in a Java virtual machine.

[2] https://jaxb.java.net/

[3] XACBecnh is a toolset developed in our research laboratory at Bu-Ali Sina University, accessible via https://github.com/nassirim/xacBench.

Table 3 Specification of synthetically generated datasets

Name	#Policies	#Rules	Type	Version
DS1_Xacml3	100	300	Categorical	Xacml3
DS2_Xacml3	100	300	Categorical-Numerical	Xacml3
DS3_Xacml3	100	100	Categorical	Xacml3
DS4_Xacml2	100	100	Categorical	Xacml2

To offer an unambiguous view of the changes, the targets of all policies in the datasets were assumed to be the same in these experiments. Also, the symmetric uncertainty threshold, $SU_{threshold}$ was set to 0.5, and the weight of every component was set to 0.33.

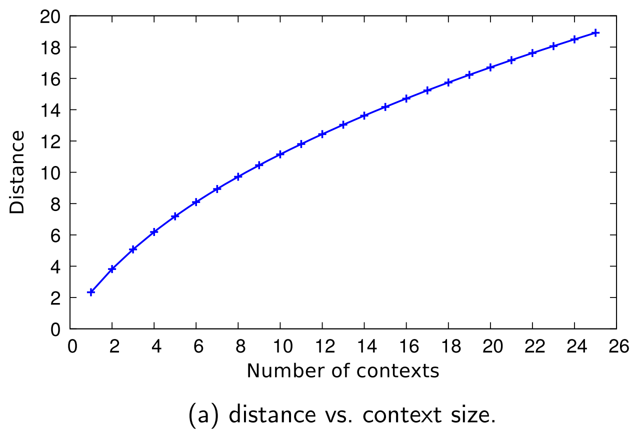
5.3 The effect of the number of context attributes

This section aims at showing the effect of the number of context attributes on distance. To this end, the dataset DS1_Xacml3 was assumed with policy pairs that have different numbers of context attributes. On receiving this dataset as its input, the proposed method calculates the distance of policies in pairs, i.e. $D(P_1, P_2)$, $D(P_3, P_4)$, $D(P_5, P_6)$, etc. In every subset of the input dataset, two distance values are obtained and averaged.

Figure 3a illustrates the relation between the distance and the number of context attributes. Every point on the plot represents one subset of the policies in the input data set. More precisely, it represents the distance between a pair of policies that have an equal number of context attributes. As shown in Fig. 3a, an increase in the number of context attributes leads to increase the distance between two policies. The increase in the distance is explained by the difference in the values of context attributes in the first policy. If increased, this difference would increase the distance. In fact, if y_k in Eqs. (17) and (18) differs from the first to the second policy, the number of dissimilar values will increase, which in turn will increase the distance.

The next experiment seeks to evaluate the effect of the number of context attributes on the degree of similarity. Again, the dataset DS1_Xacml3 was employed. XACSim calculates the similarity of policies in pairs, i.e. $S(P_1, P_2)$, $S(P_3, P_4)$, $S(P_5, P_6)$, etc. In every subset of the input dataset, two similarity values were obtained and averaged.

Figure 3b illustrates the relation between the degree of similarity and the number of context attributes. Every point on the plot represents one subset of the policies in the input data set, which have equal numbers of context attributes. As shown in Fig. 3b, an increase in the number of context attributes decreases the degree of similarity between two



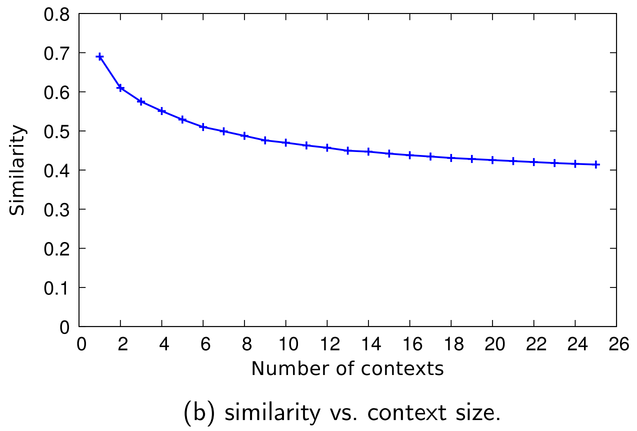


Fig. 3 Effect of the number of context attributes on distance and similarity of policies. The results were obtained based on an experiment conducted on the DS1_Xacml3 dataset. Every point on the plots represent the average distance/similarity of policy pairs with equal number of context attributes

policies, which is explained by the difference in the values of context attributes in the first policy. If increased, this difference would increase the distance and decrease similarity. In this experiment, too, if y_k in Eqs. (17) and (18) differ from the first to the second policy, the number of dissimilar values will increase, which in turn will increase the distance. As a result, the increased distance would decrease similarity.

5.4 The effect of threshold value on distance

We run another experiment to measure the effect of threshold value of *symmetric uncertainty* on distance. Here, the first 60 policies of the DS2_Xacml3 dataset were used. This dataset includes multiple policy sets, each with policies with an equal number of context attributes. XACSim takes each of these policy sets as a separate dataset and calculates the average distance between pairs of policies. The reason is that the difference in number of context attributes in these policy sets will affect our calculations.

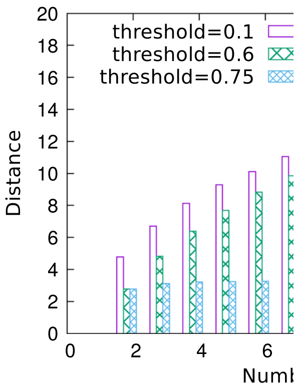


Fig. 4 The effect of threshold values on distance

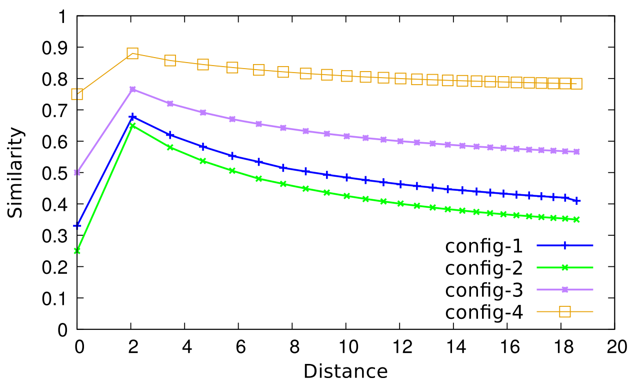


Fig. 5 The effect of different weights

Therefore, policies with identical context attributes were evaluated separately.

Figure 4 illustrates distance for a range of different threshold values. Each column represents the distance between two policies with the specified number of attributes on the *x* axis. The experiments was repeated for three different threshold values, namely, 0.1, 0.6, and 0.75. An increase in the number of attributes raises the distance for all threshold values. This increase in the distance is explained by the increase in the number of context attributes in each policy pair. The more the threshold value increases the more the distance decreases. In fact, with an increase in the threshold value, the number of context attributes will decrease. Only attributes that are most relevant to their target attribute will be selected as context attributes.

As shown in Fig. 4, the number of context attributes is increased by the increase in the number of attributes. On the other hand, the increased threshold value will reduce the number of context attributes, as indicated by equation (15). Overall, a change in the number of context attributes depends on the distribution of attributes values. Recall that an attribute with *SU* values greater than the threshold value is selected as a context attribute. According to equation (15), with an increase in the threshold value and in the number of attributes with larger *SU*, the number of context attributes will increase and, as indicated by Fig. 3a, the distance will increase.

5.5 The relation between distance and similarity

This section describes the results of an experiment that seeks to measure the effect of distance on similarity by using `DS1_Xacml3`. It separately calculates the similarity of policies in pairs, i.e. $S(P_1, P_2)$, $S(P_3, P_4)$, and the distances between pairs of policy, i.e. $D(P_1, P_2)$, and $D(P_3, P_4)$. Next, in every subset of the input data set, two similarity values (two distance values) are obtained and averaged. These average values together specify a point on the plot in Fig. 5 (`config-1`). In other words, every point on the plot signifies a pair of policies with identical context attributes, and the similarity and distance of the point on the plot refer to the average of similarity and distance calculated for the two policies. The figure also shows the effect of varying the weight of different components on the similarity score computation.

Figure 5 illustrates the inverse correlation that exists between distance and the degree of similarity. An increase in the distance would decrease the degree of similarity. The reason behind this reduction is that, when the number of context attributes with different values increases, the target attributes will be related to numerous attributes, which means that the entropy and scattering of the attributes are relatively high in the input data set and, therefore, no particular semantic attribute can be found. This is explained by Eqs. (19) and (20). An increase in distance will reduce similarity, which is indicative of the inverse relationship between distance and similarity.

Table 4 shows the weight of different components and Fig. 5 illustrates the effect of these weights on the degree of similarity. As shown in the figure, an increase in the difference between the weights of the components in each configuration will increase the difference between the degrees of similarity obtained by different configurations. In Fig. 5, for example, due to the closeness of weights in `config-1` and `config-2`, the degrees of similarity

Table 4 Weight Assignments for policy elements

Config-#	w_t	w_p	w_d
Config-1	0.33	0.33	0.33
Config-2	0.25	0.375	0.375
Config-3	0.5	0.25	0.25
Config-4	0.75	0.125	0.125

obtained by these configurations are also close to each other. Also, according to Table 4, the weights belonging to `config-4` have a greater difference, and a larger weight is allocated to the component w_t; therefore, the plot of this configuration appears to be significantly different from the other three plots in Fig. 5.

5.6 Comparison of XACSim with SPDP and PSM

This section seeks to compare XACSim with PSM and SPDP. The data set used to compare XACSim, and SPDP was `DS3_Xacml3`, and the number of requests was the product of the domain of attributes. Also, the data set used to compare the method with PSM was `DS4_Xacml2`.

Figure 6 shows the comparison results between the proposed method SPDP and PSM. The horizontal axis in this figure shows the similarity values calculated by SPDP. The vertical axis represents the similarity as calculated by PSM and XACSim. Every point in this figure indicates similarity for the first and second policy (P_1, P_2) as calculated by the three methods.

As can be seen in Fig. 6, all three mechanisms estimate similarity in the same way and with only a small amount of error for each policy pair. As SPDP uses a request evaluator to calculate the similarity between policies, it is used as a basis for comparing our results against PSM. In Fig. 6, given the bisector $y = x$, the degrees of similarity calculated by the proposed method are closer than PSM to the similarity calculated by SPDP. This is indicative of the accuracy and validity of our approach to calculate similarity.

5.7 Efficiency of the proposed method

Our last experiment run over DS2_Xacml3 to measure the execution time of each method (cf. Fig. 7).

The figure shows the elapsed time for evaluation of the policies based on the number of context attributes in each policy pair. The horizontal axis represents policies with different number of context attributes. It is observed that an increase in the number of context attributes will increase the execution time which is the sum of the elapsed times for calculation of the similarity and the distance between two policies. Every point on the plot shows the number of context attributes. This increase in time is explained by the increase in the number of context attributes. Overall, the execution time for a dataset with a large number of categorical attributes remains quite reasonable (fewer than 1 min) for both XACSim and PSM. However, PSM slightly outperforms our approach due to the hierarchical complex structure of the XACML 3.0 standard on which our mechanism is based.

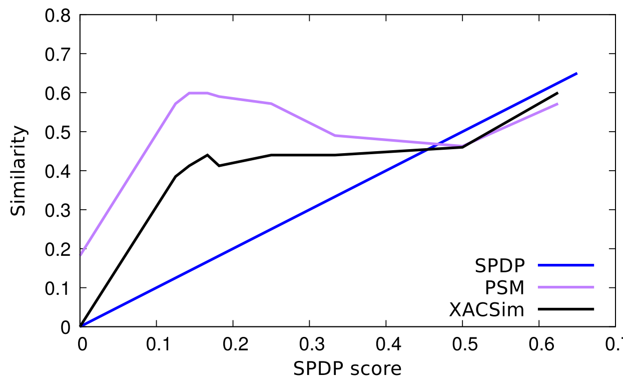


Fig. 6 Comparing XACSim vs. PSM: XACSim obtains scores closer to SPDP

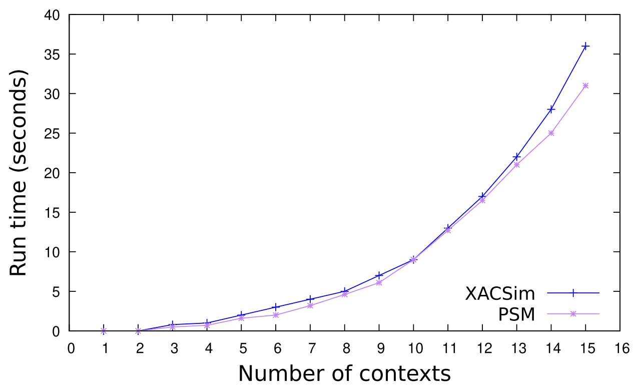


Fig. 7 The elapsed time for evaluation of similarity between policies

6 Conclusions

We proposed a hierarchical algorithm to calculate the similarity of two XACML 3.0 policies. To this end, similarity measure is calculated at four levels, i.e., value, attribute, rule, and policy. The similarity of two policies is, then, an normalized aggregate of the similarity degree at all four levels. More importantly, we developed XACSim tool based on our proposed mechanism and evaluated its effectiveness by using different XACML datasets. Results and comparisons show that our tool is able to efficiently calculate the similarity score between two XACML policies.

As a future work, we plan to use this composite similarity measure mainly for clustering policies. Although clustering may accelerate the evaluation of requests received by the PDP engine, a large number of policies in a cluster may prevent the engine from efficient analysis and turn it into a bottleneck. As a possible solution, we can define a range X-MIDD [39] for the policies in each cluster of the tree and facilitate the process of request evaluation.

Author contributions ZK: Writing-editing. MN and MR: Resources; Supervision; data analysis.

Funding This research has not been funded by any academic or industrial grant.

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest Author A, *Zahra Katebi*, declares that she has no conflict of interest. Author B, *Mohammad Nassiri*, declares that he has no conflict of interest. Author C, *Mohsen Rezvani*, declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Parducci, B., Lockhart, H., Rissanen, E.: Extensible access control markup language (XACML) version 3.0, pp. 1–154. OASIS Standard (2013)
2. Edirisinghe, M.M.: An efficient and scalable access reviw evaluation model for XACML: a subject-object graph based approach. Master's thesis, University of Moratuwa, Sri Lanka (2017)
3. Griffin, L., Butler, B., de Leastar, E., Jennings, B., Botvich, D.: On the performance of access control policy evaluation. In: 2012 IEEE International Symposium on Policies for Distributed Systems and Networks, pp. 25–32. IEEE (2012)
4. Duan, L., Zhang, Y., Chen, S., Zhao, S., Wang, S., Liu, D., Cheng, B., Chen, J., Liu, R.P.: Automated policy combination for secure data sharing in cross organizational collaborations. IEEE Access **4**, 3454–3468 (2016)
5. Vaidya, J., Shafiq, B., Atluri, V., Lorenzi, D.: A framework for policy similarity evaluation and migration based on change detection. In: International Conference on Network and System Security, pp. 191–205. Springer, Cham (2015)
6. Li, Y., Cuppens-Boulahia, N., Crom, J.-M., Cuppens, F., Frey, V., Ji, X.: Similarity measure for security policies in service provider selection. In: International Conference on Information Systems Security, pp. 227–242. Springer, Cham (2015)
7. Erradi, M.: ABAC rule reduction via similarity computation. In: Networked Systems: 5th International Conference, NETYS 2017, Marrakech, Morocco, 17–19 May 2017, Proceedings, vol. 10299, p. 86. Springer, Cham (2017)
8. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 856–863 (2003)
9. Hongxin, H., Ahn, G.-J., Kulkarni, K.: Discovery and resolution of anomalies in web access control policies. IEEE Trans. Depend. Secure Comput. **10**(6), 341–354 (2013)
10. Liu, T., Wang, Y.: Beyond scale: an efficient framework for evaluating web access control policies in the era of big data. In: International Workshop on Security, pp. 316–334. Springer, Heidelberg (2015)
11. Bertolino, A., Daoudagh, S., El Kateb, D., Henard, C., Le Traon, Y., Lonetti, F., Marchetti, E., Mouelhi, T., Papadakis, M.: Similarity testing for access control. Inf. Softw. Technol. **58**, 355–372 (2015)
12. Bhartiya, S., Mehrotra, D., Girdhar, A.: Proposing hierarchy-similarity based access control framework: a multilevel electronic health record data sharing approach for interoperable environment. J. King Saud Univ. Comput. Inf. Sci. **29**(4), 505–519 (2017)
13. Xu, D., Shrestha, R., Shen, N.: Automated strong mutation testing of XACML policies. In: Proceedings of the 25th ACM Symposium on Access Control Models and Technologies, pp. 105–116 (2020)
14. Chen, E., Dubrovenski, V., Xu, D.: Mutation analysis of NGAC policies. In: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, pp. 71–82 (2021)
15. Lin, D., Rao, P., Ferrini, R., Bertino, E., Lobo, J.: A similarity measure for comparing XACML policies. IEEE Trans. Knowl. Data Eng. **25**(9), 1946–1959 (2012)
16. Lin, D., Rao, P., Bertino, E., Li, N., Lobo, J.: EXAM: a comprehensive environment for the analysis of access control policies. Int. J. Inf. Security **9**(4), 253–273 (2010)
17. Mazzoleni, P., Crispo, B., Sivasubramanian, S., Bertino, E.: XACML policy integration algorithms. ACM Trans. Inf. Syst. Security (TISSEC) **11**(1), 4 (2008)
18. Ienco, D., Pensa, R.G., Meo, R.: Context-based distance learning for categorical data clustering. In: International Symposium on Intelligent Data Analysis, pp. 83–94. Springer, Cham (2009)
19. Marouf, S., Shehab, M., Squicciarini, A., Sundareswaran, S.: Adaptive reordering and clustering-based framework for efficient XACML policy evaluation. IEEE Trans. Serv. Comput. **4**(4), 300–313 (2010)
20. Pei, X., Yu, H., Fan, G.: Achieving efficient access control via XACML policy in cloud computing. In: SEKE, pp. 110–115 (2015)
21. Liu, A.X., Chen, F., Hwang, J., Xie, T.: Xengine: a fast and scalable XACML policy evaluation engine. In: ACM SIGMETRICS Performance Evaluation Review, vol. 36, pp. 265–276. ACM, New York (2008)
22. Ngo, C., Makkes, M.X., Demchenko, Y., de Laat, C.: Multi-data-types interval decision diagrams for XACML evaluation engine. In: 2013 Eleventh Annual Conference on Privacy, Security and Trust, pp. 257–266. IEEE (2013)
23. Ammar, N., Malik, Z., Bertino, E., Rezgui, A.: XACML policy evaluation with dynamic context handling. IEEE Trans. Knowl. Data Eng. **27**(9), 2575–2588 (2015)
24. Rezaeibagha, F., Mu, Y.: Access control policy combination from similarity analysis for secure privacy-preserved EHR systems. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 386–393. IEEE (2017)
25. Rezvani, M., Rajaratnam, D., Ignjatovic, A., Pagnucco, M., Jha, S.: Analyzing XACML policies using answer set programming. Int. J. Inf. Security **18**(4), 465–479 (2019)
26. Deng, F., Jie, L., Wang, S.-Y., Pan, J., Zhang, L.-Y.: A distributed pdp model based on spectral clustering for improving evaluation performance. World Wide Web **22**(4), 1555–1576 (2019)
27. El Hadj, M.A., Ayache, M., Benkaouz, Y., Khoumsi, A., Erradi, M.: Clustering-based approach for anomaly detection in xacml policies. In: SECRYPT, pp. 548–553 (2017)
28. Deng, F., Zhang, L.-Y.: Elimination of policy conflict to improve the pdp evaluation performance. J. Netw. Comput. Appl. **80**, 45–57 (2017)
29. Batra, G., Atluri, V., Vaidya, J., Sural, S.: Policy reconciliation and migration in attribute based access control. In: International Conference on Information Systems Security, pp. 99–120. Springer, Cham (2019)
30. Lin, L., Hu, J., Mao, X., Zhang, J.: Saphena: an approach for analyzing similarity of heterogeneous policies in cloud environment. In: 2016 IEEE 3rd International Conference on Cyber

Security and Cloud Computing (CSCloud), pp. 36–41. IEEE (2016)
31. Helil, N., Rahman, K.: Attribute based access control constraint based on subject similarity. In: 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), pp. 226–229. IEEE (2014)
32. Vijayalakshmi, K., Jayalakshmi, V.: A similarity value measure of abac security rules. In: 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), pp 565–571. IEEE, (2021)
33. Vijayalakshmi, K., Jayalakshmi, V.: Resolving rule redundancy error in abac policies using individual domain and subset detection method. In: 2021 6th International Conference on Communication and Electronics Systems (ICCES), pp. 89–96. IEEE (2021)
34. El Hadj, M.A., Khoumsi, A., Benkaouz, Y., Erradi, M.: Formal approach to detect and resolve anomalies while clustering abac policies. EAI Endorsed Trans. Security Saf. **5**(16), e3 (2018)
35. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to Information Retrieval, vol. 39. Cambridge University Press, Cambridge (2008)
36. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Elsevier, Amsterdam (2011)
37. Deepak, P., Deshpande, P.M.: Operators for Similarity Search: Semantics, Techniques and Usage Scenarios. Springer, Cham (2015)
38. Ahmadi, S., Nassiri, M., Rezvani, M.: XACBench: a XACML policy benchmark. Soft Comput. **24**(21), 16081–16096 (2020)
39. Ngo, C., Demchenko, Y., de Laat, C.: Decision diagrams for XACML policy evaluation and management. Comput. Security **49**, 1–16 (2015)

Zahra Katebi received the MSc fromBu-Ali Sina University, Iran. Her research interests includecomputer security and computer networks.

Mohammad Nassiri received the Ph.D.from Grenoble Institute of Technology (Grenoble INP), France.He is a faculty member and Associate Professor in the Facultyof Engineering, Bu-Ali Sina University, Iran. His researchinterests include wireless networking, Internet of Things,performance engineering, and computer security.

Mohsen Rezvani received the Ph.D.from UNSW Sydney, Australia. He is a faculty member andAssociate Professor in the Faculty of Computer Engineering,Shahrood University of Technology, Iran. His researchinterests include computer security, data analysis, and computernetworks.