ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 134

2nd International Conference on Control, Automation, and Artificial Intelligence (CAAI 2017)

An ACO-based Algorithm for Efficient XACML Policy Evaluation

Yunpeng Zhang[1*] and Beibei Zhang[2]

[1]Department of Information and Logistics Technology, University of Houston, Houston, U.S.
[2]College of Software and Microelectronics, Northwestern Polytechnical University, Xi'An, China
[*]Corresponding author

***Abstract*- With the explosive growth of the internet, XACML policies have grown rapidly in size and complexity, and the efficiency of ABAC decision-making is unable to meet people's increasing demands, so this paper serves to solve this problem by providing a model based on an ACO algorithm. The model first divides the XACML policy into different classifications by using an ACO algorithm, then searches for related policies by calculating the Euclidean Distance with the request attribute values and XACML policy center attribute values. This approach transforms the policy evaluation into a numerical calculation. To evaluate the efficiency and the effectiveness of these methods, the paper conducts two sets of tests. The first results shows that the classification effect of ACO algorithms is better than K-means; and the second results shows that the Euclidean Distance method is more efficient than the execution vectors used by Said Marouf, et al. to search for related policies.**

Keywords- XACML; efficient policy evaluation; ACO algorithm; euclidean distance method; ABAC

I. INTRODUCTION

A. Background

Currently, people increasingly need to access information online. Because a computer network has certain features, such as various coupling forms, uneven terminal distribution and open internet, the internet has become vulnerable to attack by hackers, and/or malware. As one of the most basic security mechanisms, access control [1] is drawing attention today, especially attribute-based access control systems (ABAC) [2]. ABAC can abstract identity, role, and resource from the traditional access control security classified information into an entity attribute. If the requester satisfies the attribute conditions defined by ABAC, they can thereby gain access permission. ABAC is the ideal model of access control policies in the open network environment and it has broad application prospects.

ABAC is a model that unifies the three basic elements - subject, object and operation - into attributes, and it has the corresponding Access Control Language XACML (eXtensible Access Control Markup Language) to provide support. XACML is the OASIS's standard language for the specification of authorization [3]. As users of information systems have expanded, XACML policies have grown rapidly in size and complexity and the challenge of controlling user access has grown. In order to improve the efficiency of access decisions, a high-performance XACML policy evaluation is needed [4, 5].

Many related studies have been proposed in recent years, some of them perform brute-force searching by comparing a request with all the rules in the XACML policy, such as Sun XACML PDP [6]. Said Marouf, et al. [7, 8] put forward an algorithm based on the K-Means clustering method and reordering techniques. On the one hand, this algorithm involved too many parameters, such that the method became complicated as the number of requests grew, and, on the other hand, the initial clustering center in the K-means Algorithm was defined artificially, resulting in inaccurate search results, which had the potential to affect the access decision result.

B. Approach

The paper proposes the method based on Ant Colony Optimization (ACO) [9] to improve the evaluation performance of XACML policy. First, the authors adopt an ACO algorithm to divide the XACML policy into different categories, then the authors search for related policies by calculating the Euclidean Distance with the request attribute values and XACML policy attribute values. This approach transforms the policy evaluation into a numerical calculation. To evaluate the efficiency and the effectiveness of the authors' methods, this paper conducts two sets of tests. The first result shows that the ACO algorithm has a better policy classing effect than K-means. The second set focuses on searching for related policies to obtain an access decision. Its result shows that the Euclidean Distance method is more efficient than the execution vectors used by Said Marouf, t al.

The rest of the paper is organized as follows. The authors first focus on XACML Language in section 2. Then the authors propose a method based on the ACO algorithm in section 3. This method involves two processes: the first step is to classify the XACML policy using the ACO algorithm and the second step is to identify related policies using the Euclidean Distance. The authors present their experimental results in section 4. And then the authors introduce related work in section 5. The authors conclude the paper in section 6.

II. XACML LANGUAGE

XACML is designed to set up a standard of access control and an authorization system. It is a flexible policy language that allows administrators to define a series of complicated access control rules to meet the demand of complex access control in real life. XACML policy sets are composed of three basic components: policy set, policy and rule. The policy set element contains a set of policies or other policy set elements

and a specified procedure for combining the results of their evaluation. The Policy element contains a set of Rule elements and a specified procedure for combining the results of their evaluation. The Rule element contains a Boolean expression that can be evaluated in isolation, but that is not intended to be accessed in isolation by PDP. In addition, The Target element identifies the set of decision requests that the parent element is intended to evaluate. The rule-combining algorithm defines a procedure for arriving at an authorization decision given the individual results of evaluation of a set of rules. Similarly, the policy-combining algorithm defines a procedure for arriving at an authorization decision given the individual results of evaluation of a set of policies.

It also supports a number of basic language extension approaches and with this property it could be applied to all kinds of complicated flexible environments. The currently developed extension and profile can make XACML work with SAML and LDAP standards in harmony.

A. *Data flow Model of XACML*

Data flow model of XACML is illustrated in Fig. I. It contains the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), the Policy administration point (PAP), and the Policy information point (PIP). A typical access control based on XACML works in the following way: When a user wants to access source protected by the system, the user sends the request to the Policy Enforcement Point (PEP) and PEP constitutes a request using subject, object, resource, action and other additional attributes. It then translates this request into XACML language by using a context processor and giving access request notification to Policy Decision Point (PDP). After PDP receives a request notification, PDP search forces more user attributes into the Policy information Point (PIP), which contains many subjects, resources, environments or other attributes. After PDP receives the feedback from PIP, it determines if the user has the right of access; and finally PEP sends a response to the user.

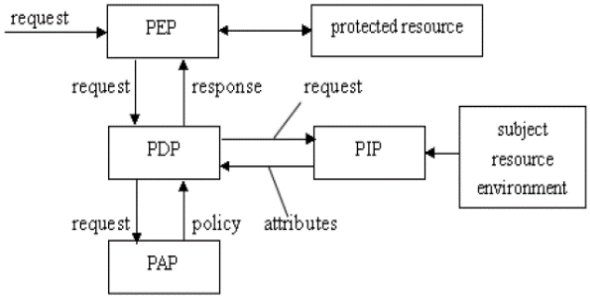

FIGURE I. DATA FLOW MODEL OF XACML

B. *Language Model of XACML*

Language model of XACML is shown in Fig. II. There are three basic elements: policy set, policy and rule. The policy set element can contain multiple policies [3]. One policy can contain multiple rules. Different rules are likely to produce conflicting results. For example, the first rule permits one request, but the second rule denies the same request, i.e. the user receives two different results. The rule combining algorithm is responsible for the problem like a judge. It will generate one final result, permit or deny. When a subject makes a request to access a resource protected by ABAC, the related policy involved in the access decision evaluation process is identified by the target element.

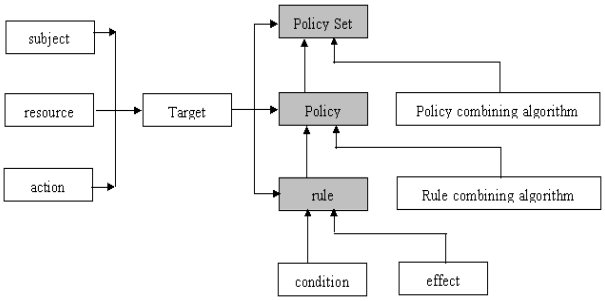

FIGURE II. FLANGUAGE MODEL OF XACML

III. THE METHOD BASED ON ACO ALGORITHM FOR EFFICIENT XACML POLICY EVALUATION

In order to improve the efficiency of XACML policy evaluation, Said Marouf, et al. [10] proposed a clustering technique that categorizes XACML policy using K-means method in respect to target subjects, then reordering these categorized policies. This method was more efficient than the standard Sun PDP. However, K-Means method has an obvious defect, because the category parameter K must be designated by the user, the bad K value would result in incorrect access decision result.

So the authors propose a method based on the ACO algorithm as shown in Fig.3. This method involves two processes: first, the authors use the ACO algorithm to classify the XACML policy according to attribute values [11], and then the authors identify related policies by the Euclidean Distance. So the system can determine whether the request is permitted or denied in less time. The policies in PIP are divided into different categories by using the ACO algorithm according to the attribute values (subjects, resources, and environments). When a user wants to access the system, PEP gives PDP access request notification, similar to the previous evaluation process. PDP searches for more user attributes in PIP, but when finding related policies, the authors calculate the Euclidean Distance with the attributes of the request and policy to reduce the duration of the policy evaluation process.

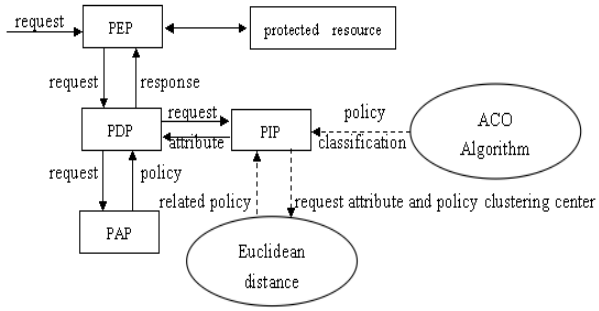

FIGURE III. FRAMEWORK OF THE METHOD BASED ON ACO ALGORITHM FOR EFFICIENT XACML POLICY EVALUATION

A. XACML Policy Categorization

In this section, the authors present a technique based on the ACO algorithm for XACML policy categorization [12, 13]. It is necessary to first offer some background on XCAML policy categorization. An ACO algorithm was introduced by Marco Dorigo, et al [14]. It is one of the most successful strands of swarm intelligence. Inspired by foraging behavior of real ants, ACO can obtain sufficient solutions within reasonable periods of time. When ants search for food, the initial state is that the ants explore the food surrounding their nest randomly. When one of them finds a food resource, it deposits different amounts of chemical pheromone along its return path and the amount of the chemical pheromone depends on the quantity and the quality of the food resource. The chemical pheromone guides other ants to the food resource and when other ants detect the existing pheromone trail, it affects their decision in choosing the paths. The more ants which choose the path, the greater the amount of the chemical pheromone which accumulates on the paths.

An ACO algorithm abstracts the main advantages existing in distributed computing, positive feedback mechanisms and greedy search for algorithms. It is not easy to lead to a local optimum situation during the search process. In the situation of the defined functional discontinuity or irregularity or noise, a greater possible optimal solution can be found.

B. Definitions and Formulas

The policy set can be expressed as a 4-tuple PS (policy set) [15]:

$$PS = (PS_{id}, PST, P, PC) \quad (1)$$

Where the policy set id (PS_{id}) reference is evaluated by resolving the reference and evaluating the referenced policy set. The policy set target (PST) specifies the set of requests to which it applies. The policy-combining algorithm (PSC) specifies the procedure by which the results of evaluating the component policy sets are combined when evaluating the policy set. The policy can be expressed as a 4-tuple P (policy):

$$P = (P_{id}, PT, R, RC) \quad (2)$$

Where the policy id (P_{id}) is evaluated by resolving the reference and evaluating the referenced policy. The policy target element specifies the set of requests to which it applies. The rule-combining algorithm (RC) specifies the procedure by which the results of evaluating the component rules are combined when evaluating the policy.

The rule can be expressed as a 4-tuple R (rule):

$$R = (R_{id}, RT, E, \text{and } C) \quad (3)$$

Where the rule id (R_{id}) reference is evaluated by resolving the reference and evaluating the referenced rule. The rule target (RT) defines the set of requests to which the rule is intended to apply in the form of a logical expression on attributes in the request. The effect (E) indicates the rule-writer's intended consequence of a "True" evaluation for the rule. Two values are allowed: "Permit" and "Deny". The condition (C) element may further refine the applicability established by the target.

This paper divides the XACML policy into different categories $(\omega_1, \omega_2, ..., \omega_m)$. The fundamental formulas needed in the algorithm are as follows:

(1)The distance d_{ij} between cluster ω_i and ω_j is calculated by:

$$d_{ij} = \| \overline{X^{(\omega_i)}} - \overline{X^{(\omega_j)}} \| = \sqrt{\sum_{k=1}^{n} (\overline{X_k^{(\omega_i)}} - X_k^{(\omega_j)})^2} \quad (4)$$

$$\overline{X^{(\omega_i)}} = \frac{1}{N} \sum_{k=1}^{N_i} X_k (X_k \in \omega_i) \quad (5)$$

Where $\overline{X^{(\omega_i)}}$ the center vectors of the cluster is ω_i, N_i is the amount of XACML policy in the cluster ω_i.

(1) The pheromone $\tau_{ij}(t)$ is calculated by:

$$\tau_{ij}(t) = \begin{cases} 1, d_{ij} \le r \\ 0, d_{ij} > r \end{cases} \quad (6)$$

Where r is the radius of attributes centers, $\tau_{ij}(t)$ is the remaining pheromone cluster ω_i to cluster ω_j at t time.

(2) Probability $P_{ij}(t)$ that cluster ω_i is merged into cluster ω_j is calculated [16] by:

$$P_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t) \eta_{ij}^{\beta}(t)}{\sum_{k=0}^{n} \tau_{ik}^{\alpha} \eta_{ik}^{\beta}(t)} \quad (7)$$

Where η_{ik} is the weight coefficient. k is the possible value of the attribute ω_i, n is the maximum possible value of the attribute ω_i, α and β are the relative variables (the author can change them) that influence the pheromone level and the heuristic value.

C. Classification Algorithm

In order to divide XACML policy into different categories, the authors take policy attributes and policy cluster centers as different ants and the food resources, respectively. The skeleton for XACML policy classification is shown as Algorithm 1. The authors first initialize the parameters of ant colonies, which belong to different categories. The ant i belongs to the category ω_i, where the ants explore the food surrounding their nest randomly. When an ant i finds a food resource ω_j (another cluster), the algorithm calculates the

distance d_{ij} between cluster ω_i and ω_j using formula (4) and (5) and updates pheromones using formula (6). The Pheromones are used to communicate with each other to find the most similar cluster for ants. Besides the ant i needs to calculate the probability cluster ω_i merged into cluster ω_j using formula (7). If the merging probability is equal to or greater than the original probability P_0, the authors update the policy cluster center for the next cycle of the algorithm.

Algorithm 1

Begin

Input<XACML policy *attributes (1: n)*>

Initialize the parameters of ant colony

While *termination conditions do not meet* ***do***

for *i=1:n-1*

*{****for*** *j=i+1:n*

Calculate d_{ij} *between cluster* w_i *and* w_j

Update pheromones

Calculate probability cluster w_i *merged into cluster* w_j

if $p_{ij}(t)>=p_0$ ***do***

Cluster w_i *merged into cluster* w_j

Update XACML policy *cluster center*

}

output<*"classification of XACML* policy*">*

end

D. *Searching for Related Policies Based on the Euclidean Distance*

In this section, the authors use the Euclidean Distance [17] to search for related policies. The paper has divided the XACML policy into M clusters, $\omega_1, \omega_2, ..., \omega_M$ .There are N_i XACML policy categories. Take cluster ω_i as an example. It can be expressed as:

$$X^{(\omega_i)} = (X_1^{(\omega_i)}, X_2^{(\omega_i)}, ..., X_{N_i}^{(\omega_i)})^T \tag{8}$$

Where $X_1^{(\omega_i)}$ expresses the first attribute value of cluster ω_i . As for access request attribute values, it can be expressed as $X = (x_1, x_2, ..., x_n)$, where x_i is the i[th] of the access request attributes. If the conditions of the formula are satisfied, there is more relevance between request X and cluster ω_j .

$$d(X, \overline{\omega_i}) < d(X, \overline{\omega_j}) \tag{9}$$

Where j=1, 2 , …, M, $i \neq j, X \in \omega_i$ the Euclidean Distance between cluster $\overline{\omega_i}$ and X is given by:

$$d(X, \overline{\omega_i}) = \sqrt{(x_1 - \overline{\omega_{i1}})^2 + (x_2 - \overline{\omega_{i2}})^2 + \cdots + (x_n - \overline{\omega_{in}})^2} \tag{10}$$

The skeleton for the related XACML policy is shown as algorithm 2. The algorithm can input the attribute values of the request and XACML policy, and then the authors calculate the Euclidean Distance between the request and each of the clusters $\overline{\omega_i}$ using formula (10). Next, the authors find the cluster that request X belongs to by comparing the relevance between request X and each of the clusters using formula (9). If the condition of formula (9) is satisfied, the request belongs to cluster ω_i . Otherwise, the request belongs to cluster ω_j .

Algorithm 2

Begin

Input<*request and* XACML policy *attribute values*>

While *termination condition* is *not satisfied* ***do***

for *i=1:n-1*

*{****for*** *j=i+1:n*

if $d(X, \overline{\omega_i}) < d(X, \overline{\omega_j})$ ***do***

Request belongs to cluster w_i

else *request belongs to cluster* w_j

}

output<*"request belongs to which cluster"*>

End

IV. Results and Discussion

The experiments were performed on a desktop PC running Windows 8 with 8G memory and Dual Core 2.66 GHz Intel processors. The paper conducts simulation with code A, B, C and D, these examples were previously used by K. Fisler and E. Martin, et al. [17, 18]. Table 1 summarizes the basic statistics of the XACML structure. The first column shows the XACML policy name, and the other columns shows policy sets, policies, and rules, respectively.

To validate this approach, the authors conducted two sets of tests. The first test case evaluated the efficiency of the XACML policy by comparing the ACO algorithm with K-means used by Said Marouf, et al. [10]; the second case discussed the effectiveness of the method using the Euclidean Distance to search for related policies.

TABLE I. ELEMENTS OF EXAMPLES

	#Policyset	#Policy	#Rule
code A	5	2	2
code B	7	3	3
code C	8	4	4
code D	11	5	5

A *XACML Policy Classification*

This paper proposed an efficient XACML Policy evaluation method first to get XACML policies classification using the ACO algorithm. In order to evaluate this method, the

authors first abstracted the attributes from code A, code B, code C, and code D XACML policy, and then expressed them in the form that is convenient for the experiment using formulas (1), (2) and (3). Then the authors divided these XACML policies into different categories using the ACO algorithm. In order to explain the superiority of their method, the authors compared the ACO algorithm and K-means used by Said Marouf, et al. [10].

Fig.IV. illustrates the categorization performance of an ACO algorithm. The authors were able to observe that these XACML policies are divided into different categories. The symbols blue '+', blue '□', light green '+', red '○' are marked as different categories.

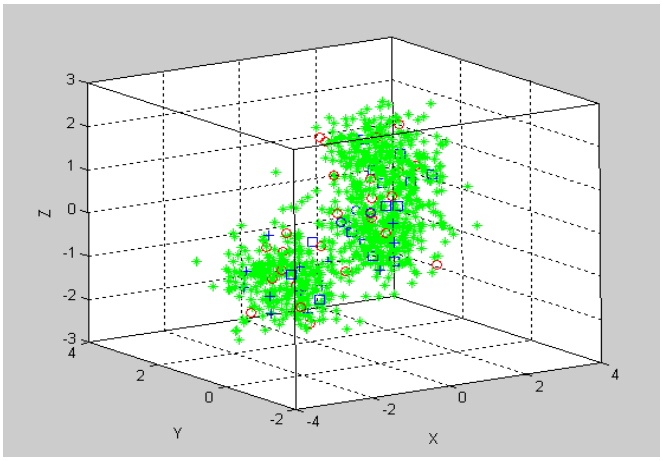

FIGURE IV. THE CATEGORIZATION PERFORMANCE OF THE ACO ALGORITHM

Fig. V. and Fig. VI. shows the categorization performance of K-means with k=3 and k=4 as shown in the two figures.

When parameter k=3, these XACML policies are divided into three different categories using K-means and the symbols blue '+', light green '+', red '○' marked as different categories.

When parameter k=4, these XACML policies are divided into four different categories using K-means and the symbols blue '+', light green '+', blue '□', red '○' marked as different categories. The observation indicates that K-means clustering has many weaknesses. For example, the number of cluster k must be determined beforehand. Furthermore, it is sensitive to different initial conditions and will produce different types of clusters, resulting in a circular cluster shape because of the distance. Thus, the ACO algorithm is superior to the k-means for classification.

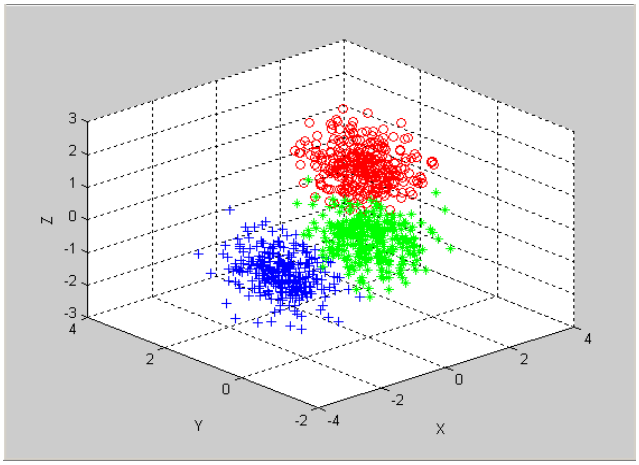

FIGURE V. THE CATEGORIZATION PERFORMANCE OF K-MEANS (K=3)

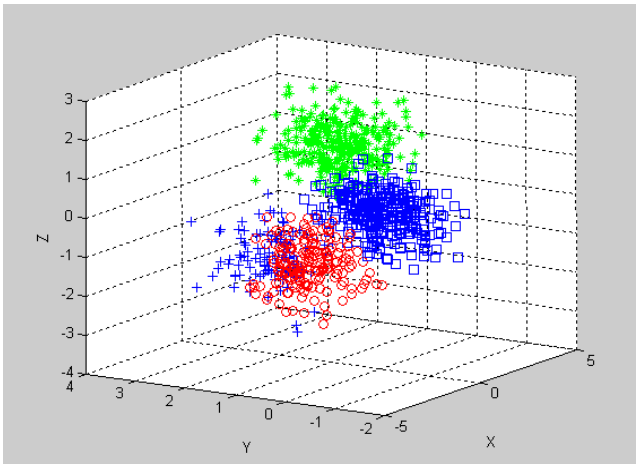

FIGURE VI. THE CATEGORIZATION PERFORMANCE OF K-MEANS (K=4)

B Effectiveness of ACO Method Using the Euclidean Distance

In this section, the authors discuss the effectiveness of the ACO method using the Euclidean Distance to search for related policies. Euclidean Distance represents the relationship between request attribute value and XACML policy cluster center. In these test cases, the authors first randomly generated request attribute values, and then searched for related XACML policies for each of them. In order to ensure the accuracy of the validation, the authors performed this step many times and obtained the access request decision using their related policy finding results. By comparing these access request decisions with the expected access request decision, the experiment proved that 100% access decision correctness can be obtained using Euclidean Distance to search for the related policies in the case of appropriate XACML policy classification.

Although the Euclidean Distance can effectively search for the XACML policy related to the request attribute to a certain extent, it can't handle a situation that involves too many attribute values. Among these attribute values, only a few may

play crucial roles, but the Euclidean Distance treats these attribute values in the same way.

V. Related Work

There is some existing research on XACML policy evaluation [4, 6, 15, 19]. Sun XACML PDP is the first and the most widely used evaluation engine and it has been implemented in Java for XML operations. It performs a brute force search by comparing a request with all the rules in an XACML policy, which calls for high-performance XACML policy evaluation engines. In order to address this problem, Kolovski, et al and S. Rizvi, et al. [20, 21] proposed two approaches to formalize XACML policies using description logics (DL), and exploited existing DL verifiers to detect and remove redundant XACML rules. These ideas may improve the performance of XACML policy evaluation to some degree, but, they become invalid when too many redundant XACML rules exist in the system.

The proposals presented by Alex X. Liu, et al. and Santiago Pina Ros, et al. have some similarities. Both of them try to improve the performance of the PDP evaluation using a tree data structure. The technique is similar to the logical expression normalization technique. XEngine [22, 23] by Alex X. Liu, et al. argues that policy normalization needs to develop a common policy language or normal form to represent policies from any application. They first normalize the given XACML policy and request and then use an efficient search feature of the tree structure, and converse normalized XACML policy to a tree structure. The latter approach builds two types of trees - Matching Tree and Combining Tree. The Matching Tree is similar to the tree built in XEngine without the numerical process. However, the big disadvantage of this approach is that it only supports the string values, such as attributes, but cannot describe the other elements perfectly, such as condition restriction of the data type.

Zhang et,al. [24] proposes a XCFG based test case selection approach for the regression testing of BPEL composite service. Different from the other methods, the work is based on the all-uses data flow testing criterion. And the proposed approach can cover the affected data flow caused by process change, binding change and interface change. Experimental study shows that the approach has a high coverage rate for the changes. Besides test case selection, the approach can also be used for the all-used criterion based test case generation for BPEL composite service of single versions. However, there are a lot of works to do in the future, such as performing more large-scale experimental study, using the proposed approach to Web composite service described in other languages, and so on.

Reference [25] applies ABC based algorithm for test data generation, which is an NP-hard problem and conventional methods are not given accurate results for such type of problems. In order to generate the optimize suite of test data and also to overcome limitations of conventional methods, the paper adopts an ABC based algorithm deal this problem. ABC algorithm is characterized by the bee behavior and this algorithm has strong exploration and exploitation capabilities among same class of algorithms. In this work, ten benchmark problems of different dimensions are used to examine the performance of the proposed ABC algorithm. The performance of the proposed algorithm is measured on two performance matrices and also compared other well know algorithms exist in literature such as random search, GA and PSO for data-flow coverage.

The work of [10] is similar to the method proposed by Said Marouf, et al., in which the authors propose a two-step framework for the efficiency of the PDP evaluation. In other words, they categorize policies and rules within a policy set and policy, respectively, with respect to target subjects using the K-Means algorithm in the first step; then they dynamically optimize the ordering of policies within a policy set and rules within a policy to obtain a related XACML policy. However, the K-Means method that been utilized for XACML policy classification, the initial clustering center is defined subjectively. This clustering center is very important. If it is defined artificially, it probably results in a bad clustering result, which leads to incorrect access decision result.

VI. Conclusion

This paper describes a framework for an ACO-based algorithm for efficient XACML Policy evaluation and makes two key contributions. First, the authors adopt an ACO algorithm to divide the XACML policy into different classifications. Second, to search for the related XACML policies, the authors use the Euclidean Distance method to calculate the distance between the request attribute values and XACML policy center attribute values. When an access request is received, the system first calculates the distance between the attribute value of the given request and the attribute cluster which has already been classified by the ACO algorithm using the Euclidean Distance method to search for the XACML policies related to the request, so that the authors transform the related policy evaluation into a numerical calculation. To evaluate the efficiency and the effectiveness of their methods, the authors conducted two sets of tests. The first results show that the classification effect of an ACO algorithm is better than K-means. The second results show that the Euclidean Distance method is more efficient than the execution vectors used by Said Marouf, et al. to search for related policies.

Acknowledgment

The research was funded by the University of Houston - National Research University Fund (NRUF). The project number is 110664 LINKED TO 105433.

References

[1] R. Kuhn et al., Information Technology-Role Based Access Control, ANSI Standard, Document Number: ANSI/INCITS 359-2004, Feb. 03, 2004

[2] E. Yuan, J. Tong., Attributed based access control (ABAC) for Web services, ICWS 2005. Proceedings. IEEE International Conference on Web Services, pp. 569, 11-15 July 2005

[3] eXtensible Access Control Markup Language (XACML) Version 3.0 core specification[S] http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cos01-en. pdf

[4] S. P. Ros, M. Lischka, F. G. Mármol. Graph-based XACML evaluation. In Proceedings of the 17th ACM symposium on Access Control Models and Technologies (SACMAT '12). ACM, New York, NY, USA, 83-92.

[5] H. Tao, A XACML-based Access Control Model for Web Service, International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1140-1144, 2005.

[6] Sun's XACML open resource implementation. http://sunxacml. resourceforge.net. June, 2015

[7] M. Chung, S. Chen, C. Yu, P. Chen, An improvement of fast search for algorithm for vector quantization, Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on, pp. 97-100, 13-16 Dec. 2005

[8] R. Rivest, On Self-Organizing Sequential Search for Heuristics, Comm. ACM, vol. 19, no. 2, pp. 63-67, 1976.

[9] M. Dorigo, M. Birattari. Ant colony optimization [M] //Encyclopedia of Machine Learning. Springer US, 2010: 36-39.

[10] S. Marouf, M. Shehab, A. Squicciarini, et al. Statistics & clustering based framework for efficient XACML policy evaluation[C]// policy sets for Distributed Systems and Networks, 2009. policy2009. IEEE International Symposium on. IEEE, 2009: pp.118-125.

[11] D. Lin, P. Rao, E. Bertino, and J. Lobo. An approach to evaluate policy similarity. In SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies, pp. 1–10, New York, NY, USA, 2007.

[12] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino, XACML policy Integration Algorithms, ACM Trans. Information and System Security, vol. 11, no. 1, Feb. 2008

[13] D.E. Taylor, Survey & Taxonomy of Packet Classification Techniques, ACM Computing Surveys, vol. 37, no. 3, pp. 238-275, 2005.

[14] M. Dorigo, Marco Birattari and Thomas Stutzle, Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique, IEEE Computational Intelligence Magazine, Vol. 1, No. 4, November 2006.

[15] P. L. Miseldine. Automated XACML policy reconfiguration for evaluation optimization. Proceedings of the ftheirth international workshop on Software engineering for secure systems SESS 08, pages 1–8, 2008.

[16] K. Frikken, M. Atallah, and J. Li, Attribute-Based Access Control with Hidden Policies and Hidden Credentials, IEEE Trans. Computers, vol. 55, no. 10, pp. 1259-1270, Oct. 2006.

[17] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change impact analysis of access-control policies. In Proc. 27th International Conference on Software Engineering, pages 196 ‑205, 2005.

[18] E. Martin, T. Xie, T. Yu. Defining and measuring policy coverage in testing access control policies [M]//Information and Communications Security. Springer Berlin Heidelberg, 2006: 139-158.

[19] F. Turkmen and B. Crispo. Performance evaluation of XACML PDP implementations. Proceedings of the 2008 ACM workshop on Secure The author sb Services, pages 37–44, 2008.

[20] V. Kolovski and J. Hendler, XACML policy Analysis Using Description Logics, http://www.mindswap.org/~kolovski/ Kolovski XACML AnalysisJCS Submission.pdf, August 2014.

[21] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In Proceedings of the International Conference on Management of Data, pages: 551–562, New York, NY, USA, 2004. ACM

[22] A. X. Liu. F. Chen, H. JeeHyun, T. Xie, Designing Fast and Scalable XACML policy Evaluation Engines, Computers, IEEE Transactions on, vol. 60, no. 12, pp. 1802, 1817, Dec. 2011

[23] A. X. Liu, F. Chen, J. H. Hwang, et al. Xengine: a fast and scalable XACML policy evaluation engine[C] //ACM SIGMETRICS Performance Evaluation Review. ACM, 2008, 36(1): 265-276.

[24] Ji S, Li B, Zhang P. Test Case Selection for Data Flow Based Regression Testing of BPEL Composite Services[C]. IEEE International Conference on Services Computing. IEEE, 2016:547-554.

[25] Kumar S, Yadav D K, Khan D A. Artificial Bee Colony based Test Data Generation for Data-Flow Testing[J]. 2016, 9(39)