Mining Attribute-based Access Control Policies by Taking Aspirations form Particle Swarm Optimization Algorithm

Masoud Narouei*, Hassan Takabi*, and Laura Adams†
***Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA**
† Texas State University, San Marcos, Texas , USA

Abstract

ABAC has been emerged as a general model that could overcome the limitations of the dominant access control models (i.e., MAC, DAC, and RBAC) while unifying their advantages. Once fully implemented within an enterprise, ABAC promotes information sharing while maintaining control of the information. However, ABAC's policy specification is more complex and can result in an expensive and time consuming task. To reduce this burden, policy mining algorithms have been proposed to partially or totally automate the construction of ABAC policies from available access control information. In this work, we propose a nature-inspired methodology for solving the policy mining task. Initial results proved the effectiveness of the proposed methodology.

Introduction

Attribute-based access control (ABAC) is an access control model wherein the access control decisions are made based on a set of attributes, associated with the requester, the environment, and/or the resource itself. ABAC allows ``an unprecedented amount of flexibility and security while promoting information sharing between diverse and often disparate organizations''. ABAC promises long-term cost savings through reduced management effort, however, **manual development of ABAC initial policies can be difficult and expensive**. A promising approach to diminish the burden of policy specification is represented by policy mining, whose goal is to partially or totally automate the construction of an ABAC policy from available access control information (e.g., access control logs, RBAC policies).

Research Goal

We propose a nature-inspired methodology for learning ABAC policies from sets of authorized and denied access requests and attribute data.

Method

Considering the following access request which means faculty can add scores for cs601 course:

Request = <Faculty, addScore, cs601>

And the following attributes:

faculty = <position=faculty, department=cs, crsTaught={cs601}>
cs601 = <department=cs, crs=cs601, type=gradebook>

Our methodology looks for an ABAC policy in the following form that satisfies the request:

<type ϵ {gradebook}, {readScore, addScore}, crs ϵ crsTaught>

Algorithm 1 Particle Swarm Optimization

```
1:  procedure PSO
2:      Initialize particle population using requests
3:      Find initial global best particle
4:  LOOP:
5:      FOR each particle:
6:      Decide if it will be updated based on global best position,
7:  previous best position or local best position.
8:      if position is updated then
9:          if new position is better than current rule: then
10:             replace current rule with updated rule
11:         if new position is better than previous best rule: then
12:             replace previous rule with updated rule
13:         if new position is better than global best rule: then
14:             replace previous rule with updated rule
15:     end FOR.
16:     update global best rule
17:     goto LOOP.
```

Experiments

We evaluated our proposal experimentally on University.abac case study considered in previous research. This case study consists of a set of 22 users, 34 resources, 9 operations, and 10 rules. Users and resources are associated with various attributes. Rules were carefully constructed to express non-trivial policies and exercise all the features of the policy language, including use of set membership and superset relations in attribute expressions and constraints. We executed our approach on this case study and were able to **reduce the number of generated rules** to 7. WSC of an ABAC policy is a weighted sum of the number of elements in the policy. We use WSC to measure the complexity of generated rules. Generally less complex rules are more favorable. **Another important result is that our approach tends to generate a policy which is less complex than the baseline (WSC was reduced by 4).**

Results

Case study	$\|U\|$	$\|R\|$	$\|O\|$	$\|A_U\|$	$\|A_R\|$	$\|S_A\|$	$\|S_D\|$	$\|P_0\|$	$WSC(P_0)$	$\|P_1\|$	$\|WSC(P_1)\|$
University	22	34	9	6	5	168	6,564	10	37	7	33

- $|U|$ is number of users
- $|R|$ is number of resources,
- $|O|$ is number of operations
- $|A_U|$ is number of user attributes
- $|A_R|$ is number of resource attributes
- $|S_A|$ is number of requests accepted
- $|S_D|$ is number of requests denied
- $|P_0|$ is number of original rules
- $|P_1|$ is number of newly generated rules
- **WSC** is weighted sums of the complexity

References

[1] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, and others. 2013. Guide to Attribute based access control (ABAC) definition and considerations (draft). NIST special publication 800, 162 (2013).

[2] Eric Medvet, Alberto Bartoli, Barbara Carminati, and Elena Ferrari. 2015. Evolutionary inference of Attribute-based access control policies. In International Conference on Evolutionary Multi-Criterion Optimization. Springer, 351–365.

[3] Zhongyuan Xu and Scoott D Stoller. 2015. Mining Attribute-based access control policies. IEEE Transactions on Dependable and Secure Computing 12, 5 (2015), 533–545