HyPE: A Hybrid Approach toward Policy Engineering in Attribute-Based Access Control

Saptarshi Das[1], Shamik Sural[1], *Senior Member* Jaideep Vaidya[2], *Senior Member* and Vijayalakshmi Atluri[2], *Senior Member*
[1]Indian Institute of Technology, Kharagpur, India
[2]Rutgers University, Newark, NJ, USA
saptarshidas13@iitkgp.ac.in,shamik@cse.iitkgp.ac.in, jsvaidya@rutgers.edu, atluri@rutgers.edu

Abstract—Successful deployment of attribute-based access control requires the process of policy engineering which involves constructing a set of appropriate rules, known as a policy. Policy engineering is performed either by a top-down approach that may ignore some of the existing accesses in the organization or a bottom-up approach that may form rules which are not relevant to the organizational processes. In this work, we propose a hybrid approach toward policy engineering that addresses the limitations of the top-down and the bottom-up approaches while preserving their individual advantages.

Index Terms—Attribute-based access control, Policy engineering, Hybrid approach, Entropy.

1 Introduction

In recent years, Attribute-Based Access Control (ABAC) [8] has emerged as the desired access control model for many organizations. Although contemporary access control models like Role-Based Access Control (RBAC) [15] suffice in situations involving a known set of subjects, they are not that effective in dynamic situations, i.e., scenarios where resource sharing among organizations is unexpectedly necessary, which is where ABAC shines. This necessitates organizations to migrate to ABAC for which an ABAC policy is essential. In ABAC, a subject's request to perform an operation on an object is permitted or denied based on the attributes of the requesting subject, the requested object, the environment in which the request is made, and a policy (set of rules) specified in terms of those attributes. In other words, ABAC policies mediate access to objects by evaluating rules against the attributes of the requesting subject and the requested object, operations, and the environment to a request. Each such permitted request is called an access.

It may be noted that, before the advent of ABAC, RBAC served as the workhorse access control model instantiated in various forms. For successful deployment of RBAC, an appropriate set of roles had to be identified using a process commonly referred to as role engineering. Likewise, defining an appropriate policy is crucial for successful deployment of ABAC. This process, known as *policy engineering* [9], has been identified as one of the most difficult and costliest components in implementing ABAC [6]. Policy engineering is similar to the problem of role engineering in RBAC and like role engineering, there are two basic approaches to policy engineering, namely, top-down and bottom-up.

In the top-down approach, the organizational processes are decomposed into functionally independent units by discussions with the various organizational authorities. These functional units are then associated with accesses, from which the rules are constructed. Although the formed rules provide a perspective on the organizational processes, this approach is not scalable and requires extensive human intervention. Interestingly, the issues of scalability and extensive human intervention are also endemic to the top-down role engineering approaches in RBAC [13] and, therefore, impair their usability. In contrast, the bottom-up approach, also called policy mining, takes into account the existing accesses for constructing the rules. While the accesses already present in the organization are preserved, the formed rules often tend to overlook the organizational business processes, thus making it difficult for employees to comprehend the same. It may be noted that the aforementioned shortcoming of ABAC policy mining is similar to that of role mining in RBAC where the formed roles may not reflect the organizational processes [13]. For large organizations, identifying and decomposing all the organizational processes, as well as obtaining all the existing accesses is arduous. Therefore, depending exclusively on either a top-down or bottom-up approach is not effective.

Manuscript submitted: 20-Oct-2018; Manuscript revised 27-Nov-2018; Final manuscript received: 24-Dec-2018.

Every organization has a governance body (IT team), members of which are responsible for the management of all identity, credential as well as access management capability deployment and operation. We refer to the members of such a governance body as Security Officers (SOs). It is expected that the security officers (either individually or collectively) can provide a *yes* or *no* answer when asked if a given subject is permitted to perform a specific operation on an object. Obviously, this may in specific cases, require consultation of an organization wide policy, consultation with peers or managers, domain knowledge, or other constraints. However, such consultation, if required, can be done off-line and essentially, we assume that the SO has the ability to provide a clear answer for all possible cases.

With the above context in mind, we propose a hybrid approach for policy engineering in ABAC which first uses a top-down approach to choose an access relevant to the organizational processes and ask the SO whether the chosen access is permitted or denied and then, using a bottom-up approach, forms the rules using the accesses resolved by the SO. This procedure is used iteratively until all the organizational processes are covered and a complete ABAC policy is constructed. The accesses to be resolved by the SO are chosen based on the entropy of an attribute. Entropy is the measure of the information content of each attribute. Attributes with higher entropy are more likely to influence the access decisions. Using such a metric for selecting accesses minimizes the number of accesses resolved by the SO for constructing the ABAC policy.

2 Preliminaries

An ABAC system consists of a set of subject attributes (S_a), object attributes (O_a), environment attributes (E_a) and possible operations (OP). Each attribute $a \in S_a \cup O_a \cup E_a$ has a set (AV_a) of possible values. The attributes and their assigned values for all subjects are represented as a matrix (M_s) where, each row and column represents a subject and an attribute, respectively. Each element of M_s contains a value assigned to an attribute. The matrices M_o and M_e are similarly represented for object attribute-value matrix and environment attribute-value matrix, respectively. In this work, we consider only one operation named $access$. The current work can easily be extended for multiple operations. Each rule is denoted by a 4-tuple ⟨*SC*, *OC*, *EC*, *OP*⟩, where *SC*, *OC*, *EC* and *OP*, respectively represent a set of subject attribute-value pairs, object attribute-value pairs, environment attribute-value pairs and an operation.

Generally, numerous subject and object attributes are available in an organization, but the majority of it seldom influences the access decisions. Usually, the attributes which have more number of possible values and the occurrence of those values are evenly distributed are more likely to influence access decisions. For instance, a university offers more number of courses to the students as compared to the number of departments. If we consider the projects assigned to students as objects, the assignment of projects to students usually depends on the course undertaken by the student rather than the department to which the student belongs. Attributes with a large number of possible values contain more information as compared to other attributes. Shannon Entropy [16] or simply entropy is used to quantify the information content of probability distributions. In this work, we compute the entropy of an attribute using the probability distribution of occurrence of its possible values. The entropy (H) of a given attribute X is computed as follows:

$$H_X = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \qquad (1)$$

where, $x_1, x_2, ..., x_n$ are the possible values of X and $p(x_i)$ is the fraction of subjects having the value x_i for attribute X.

3 Hybrid Policy Engineering Problem

In this section, we formally define the hybrid policy engineering (HyPE) problem.

3.1 Problem Definition

As discussed in Section 1, policy engineering is crucial for deploying ABAC in an organization. We also saw that both the top-down and bottom-up approaches used for policy engineering have their own limitations. In an organization, many employees have similar attribute values. Therefore, resolving an access involving a given subject s enables us to resolve all the accesses involving all the subjects which have the same attribute values as s. In this context, we propose the HyPE problem as defined below.

Definition 3.1. HyPE
Given a set S of subjects, set O of objects, set S_a of subject attributes, set O_a of object attributes, a subject attribute-value matrix M_s, an object attribute-value matrix M_o and a set OP of operations, construct an ABAC policy P by repeatedly resolving accesses from the SO in such a way that no extraneous access is permitted by P and the number of accesses resolved by the SO is minimum.

Thus, solving HyPE enables an organization to deploy ABAC by constructing a policy. An overarching requirement is that the constructed policy must satisfy all the existing accesses and no extraneous access must be permitted.

3.2 Complexity Analysis

Here, we do a formal analysis of the complexity of HyPE. We show that the problem is NP-Complete. To initiate the proof, we first formulate a decision version of HyPE.

Definition 3.2. Decision Version of HyPE (D-HyPE)
Given a set S of subjects, set O of objects, set S_a of subject attributes, set O_a of object attributes, a subject attribute-value matrix M_s, an object attribute-value matrix M_o and a set OP of operations, is it possible to construct an ABAC policy P by repeatedly resolving accesses from the SO in such a way that no extraneous access is permitted by P and the number of accesses resolved by the SO is $\leq k$?
To prove that D-HyPE is NP-Complete, we utilize a known NP-Complete problem, namely, the Minimum Set Cover (MSC) [7] problem, which is defined below.

Definition 3.3. Minimum Set Cover (MSC) Problem
Given a universal set U and a collection ST of subsets of U, find a minimum number of subsets $st_1, st_2, ..., st_m$ where, each $st_i \in ST$ and $st_1 \cup st_2 \cup ... \cup st_m = U$.

Definition 3.4. Decision Version of MSC (D-MSC) Problem
Given a universal set U and a collection ST of subsets of U and an integer t, does there exist a collection of subsets $st_1, st_2, ..., st_m$ which covers all the elements in U and $m \leq t$?

Theorem 1. D-HyPE is NP-Complete.
We prove that D-HyPE is NP-Complete by first showing that D-HyPE is in NP, followed by a reduction of D-MSC to D-HyPE in polynomial time. For the proof, we consider the SO as an entity which, when given an access, tells whether the access is permitted or denied.

Let A be the set of all possible $|S| \times |O|$ accesses in the organization. Each access resolved by the SO covers all the accesses that have the same associated attribute-value pairs. Let A_c be a collection which contains sets of accesses corresponding to each access resolved by the SO. Given a certificate consisting of a set of accesses A, a collection A_c with each element $a_c \in \mathcal{P}(A)$ (the power set of A) and a collection A_s such that $A_s \subseteq A_c$, where all the given accesses in each element of A_s are resolved by the SO. It can be verified in polynomial time whether $|A_s| \leq k$ by counting the number of elements of A_s. Further, it can be proved that A and A_c are consistent with A_s by verifying that all the subsets of accesses in A_s are also present in A_c and the union of all the subsets of accesses in A_s contains all the accesses in A. Thus, D-HyPE is in NP.

Now, we prove that D-MSC $\leq_p$ D-HyPE. Let $< U, ST, t >$ be an input instance of D-MSC. Then, an instance of D-HyPE can be generated in polynomial time by the assignments $A = U$, $A_c = ST$ and $k = t$. Here, A is the set of all possible accesses and A_c is the set of subsets of A. Now, let H and A_s be the solutions to the instances of D-MSC and D-HyPE, respectively. To conclude the proof, we show that the output instance of D-MSC, i.e., H is such that $|H| \leq t$, if and only if the output instance of D-HyPE, i.e., A_s is such that $|A_s| \leq k$.

Now, the solution to the instance of D-HyPE can be acquired from H as follows: Since $U = A$, each element in U can be mapped to an access in A and each subset of elements in ST can be mapped to a set of accesses in A_c. Now, if the union of all the elements of H contains all the elements in U, then the union of all the subsets of accesses in A_s is bound to contain all the accesses in A. Thus, the solution to the instance of D-HyPE obtained from the solution to the instance of D-MSC is a correct solution. The converse also holds in exactly the same way. Thus, D-HyPE is NP-Hard.

Since D-HyPE is in NP and is NP-Hard, it is NP-Complete. Note that this essentially means that we cannot expect to get an exact solution in polynomial time. One possible approach to get an exact solution is to use integer linear programming (ILP). In fact, such an approach has been formulated in the past for role mining [12], where a candidate set of roles is first identified and then the ILP chooses the optimal set from among them. In our case this corresponds to finding the candidate set of access resolution requests and choosing from among them. If all possibilities are to be considered, the candidate space grows exponentially, which is not scalable. Otherwise, the effectiveness of the approach depends crucially on how the candidates are generated. While the ILP approach could be one alternative, it is also possible to create an effective greedy heuristic, which is presented below.

3.3 Heuristic approach for solving HyPE

Algorithm 1 presents a greedy heuristic that takes a subject attribute-value matrix M_s, an object attribute-value matrix M_o, a set of subject attributes S_a, a set of object attributes O_a as inputs and returns an ABAC policy as output by resolving accesses with the help of a SO.

Algorithm 1: $HyPE(M_s, M_o, S_a, O_a)$

```
1  $SA_e \leftarrow$ sub. attr. sorted w.r.t their entropies
2  $OA_e \leftarrow$ obj. attr. sorted w.r.t their entropies
3  $SA_v \leftarrow$ sub. attr. values sorted w.r.t. their frequencies
4  $OA_v \leftarrow$ obj. attr. values sorted w.r.t. their frequencies
5  $Q_{so} \leftarrow 0$, policy $\leftarrow [\,]$, $D_i \leftarrow [\,]$, $Q_u \leftarrow [\,]$
6  $Q_i \leftarrow$ rules consisting of sub. attr. with highest entropy and
      value with lowest frequency and obj. attr. with highest
      entropy and value with highest frequency
7  for $i \leftarrow 1$ to $|Q_i|$ do
8      $Q_{so} \leftarrow Q_{so} + 1$
9      if $SO(Q_i[i] == ND)$ then
10         $D_i \leftarrow D_i \cup ND$
11         $Q_u \leftarrow Q_u \cup Q_i[i]$
12     end
13     if $SO(Q_i[i] == Y)$ then
14         $D_i \leftarrow D_i \cup Y$
15         policy $\leftarrow$ policy $\cup Q_i[i]$
16     end
17 end
18 for $a \leftarrow 2$ to $|S_a|$ do
19     $Q_i^{'} \leftarrow [\,]$
20     for $i \leftarrow 1$ to $|Q_u|$ do
21         for $j \leftarrow 1$ to $|SA_v[SA_e[a]]|$ do
22             $Q_u[i][SA_e[a]] = SA_v[SA_e[a]][j]$
23             for $k \leftarrow 1$ to $|OA_v[OA_e[a]]|$ do
24                 $Q_u[i][OA_e[a] + |S_a|] = OA_v[OA_e[a]][k]$
25                 $Q_i^{'} \leftarrow Q_i^{'} \cup Q_u[i]$
26             end
27         end
28     end
29     $D_i^{'} \leftarrow [\,]$ ; $Q_u^{'} \leftarrow [\,]$
30     for $i \leftarrow 1$ to $Q_i^{'}$ do
31         $Q_{so} \leftarrow Q_{so} + 1$
32         if $SO(Q_i[i] == ND)$ then
33             $D_i^{'} \leftarrow D_i^{'} \cup ND$
34             $Q_u^{'} \leftarrow Q_u^{'} \cup Q_i^{'}[i]$
35         end
36         if $SO(Q_i[i] == Y)$ then
37             $D_i^{'} \leftarrow D_i^{'} \cup ND$
38             policy $\leftarrow$ policy $\cup Q_i^{'}[i]$
39         end
40     end
41     $Q_i = Q_i^{'}$ ; $Q_u = Q_u^{'}$
42     if $ND$ not in $D_i^{'}$ then
43         break
44     end
45 end
46 post_process(policy)
47 return $Q_{so}$, policy
```

Step 1: Compute the entropy of each attribute
This step (Lines 1-2) computes the entropies of all the subject attributes using equation 1, and sorts them in non-increasing order of their entropies. It returns an ordered list consisting of entropies of all the subject attributes. Similarly, the entropies of all the object attributes are also computed. The worst case time complexity of this step is $O(|S_a| + |S_a|log|S_a| + |O_a| + |O_a|log|O_a|)$.

Step 2: Compute the frequency of each attribute value
In the second step (Lines 3-4) of Algorithm 1, we first find the frequencies of all the attribute values. Then we sort the frequencies of all the subject attribute values in non-decreasing order and object attribute values in non-increasing order. The worst case time complexity of this step is $O(|S||O| + |S_a|(|S_{av}|log|S_{av}|) + |O_a|(|O_{av}|log|O_{av}|))$.

Step 3: Form the rules of the policy
In the third step (Lines 5-45) of Algorithm 1, a subject attribute-value pair is selected in such a way that the attribute has the highest entropy and the value has the lowest frequency. Next, an object attribute-value pair is selected such that the attribute has the highest entropy and the value has the highest frequency. The subject and object attributes with the highest entropies are selected because they are more likely to influence the access decisions. For the selected subject attribute, the value with the lowest frequency is selected which is associated with a few subjects. While, for the object attribute, the attribute value with the highest frequency is chosen which is associated with more number of objects. As only a few subjects in an organization are permitted to access a large number of objects, such a selection criterion generates meaningful rules with less number of visits to the SO. Then, the two selected attribute-value pairs are given to the SO. If all the subjects having the chosen subject attribute-value pairs can access all the objects having the chosen object attribute-value pairs, the SO resolves such an access as yes (Y) and as no (N) otherwise. If the SO fails to resolve the access, it returns a $not\ decided$ (ND). When the SO resolves an access as Y, all the accesses corresponding to the subjects and objects which have the same subject attribute-value pairs and object attribute-value pairs, respectively, are set to Y. Likewise, if the SO resolves an access as N, all the accesses corresponding to the subjects and objects which have the same subject attribute-

value pairs and object attribute-value pairs, respectively, are set to N. The subject and object attribute-value pairs for which the SO returns a Y are formed into a rule.

The subject and object attributes are selected in a cumulative manner. First, one subject attribute and one object attribute is selected. The values of the chosen attributes are selected in increasing and decreasing order of their frequencies, for subject attributes and object attributes, respectively. This has a worst case time complexity of $O(|S_{av}||O_{av}|)$. When all the selected attribute-value pairs are resolved by the SO but there are unresolved accesses, two subject, and object attribute-value pairs are selected in a similar manner. Resolving accesses for two subject and object attributes has a worst case time complexity of $O(|S_{av}|^2|O_{av}|^2)$. This procedure is repeated until all the accesses are resolved. Finally, post-processing is performed on the formed rules.

Step 4: Post-processing

In this step (Line 46) of Algorithm 1, the rules obtained so far are simplified wherever possible. For instance, if there are two rules with similar set of subject attribute-value pairs, we form a single rule by merging the object attribute-value pairs of the rules, by adding multiple values to the object attributes. For instance, if $r_1 =< \{S.DESG = PROF; S.DEPT = CSE\}, \{O.TYPE = TND; O.DEPT = CSE\}, access >$, and $r_2 =< \{S.DESG = PROF; S.DEPT = CSE\}, \{O.TYPE = TND\}\{O.DEPT = FIN\}, access >$ with their subject parts same, we form a single rule r_{12} by merging the object parts of the rules r_1 and r_2 where, $r_{12} =< \{S.DESG = PROF; S.DEPT = CSE\}, \{O.TYPE = TND; O.DEPT = CSE, FIN\}, access >$.

3.4 Illustrative Example

Now, we illustrate our proposed solution using an example. Let us consider an university having two departments, namely, Computer Science and Engineering (CSE) and Civil Engineering (CE). Each member of the college is associated with a designation ($S.DESG$) among Student (STU) and Professor ($PROF$). The subjects belong to one of the aforementioned departments ($S.DEPT$). The objects in the organization include Assignment ($ASGN$), Tender (TND) and Attendance Log ($ATTL$). Each object also belongs to a specific department ($O.DEPT$) among the ones mentioned previously.

Now, we try to form a policy using our proposed heuristic solution to HyPE. First, the entropies of all the subject and object attributes are computed. Let us consider how the entropy of the attribute $S.DESG$ is computed using Equation 1. In Table 1, the subject attribute $S.DESG$ has 2 possible values, namely, STU and $PROF$. If a subject is chosen randomly, the probability of it having the value STU associated with the attribute $S.DESG$ is $\frac{1}{2}$, i.e., $P(STU) = \frac{1}{2}$. Similarly, $P(PROF) = \frac{1}{2}$. Computation of entropy for $S.DESG$ is given below.

$$\begin{aligned} H_{S.DESG} &= -P(STU)log_2P(STU) - P(PROF)log_2P(PROF) \\ &= -\frac{1}{2}log_2(\frac{1}{2}) - \frac{1}{2}log_2(\frac{1}{2}) \\ &= 1.0 \end{aligned}$$

Similarly, the entropies for the attributes $S.DEPT$, $O.DEPT$ and $O.TYPE$ are computed as 1.0, 1.0 and 1.5, respectively.

The accesses along with the decisions of the SO are given in Table 2 where, the highlighted rows represent the formed rules. Here, out of 16 possible accesses, the SO resolves only 10 accesses to form an ABAC policy.

TABLE 1: Entities with their attribute-value assignments

S	$S.DESG$	$S.DEPT$	O	$O.TYPE$	$O.DEPT$
s_1	STU	CSE	o_1	$ASGN$	CSE
s_2	STU	CE	o_2	$ASGN$	CE
s_3	$PROF$	CSE	o_3	$ATTL$	CSE
s_4	$PROF$	CE	o_4	TND	CE

TABLE 2: Accesses resolved by the SO and their decisions

P	$S.DESG$	$S.DEPT$	$O.TYPE$	$O.DEPT$	$DEC.$
r_1	STU	–	$ASGN$	–	ND
r_2	STU	–	$ATTL$	–	N
r_3	STU	–	TND	–	N
r_4	$PROF$	–	$ASGN$	–	Y
r_5	$PROF$	–	$ATTL$	–	Y
r_6	$PROF$	–	TND	–	Y
r_7	STU	CSE	$ASGN$	CSE	Y
r_8	STU	CSE	$ASGN$	CE	N
r_9	STU	CE	$ASGN$	CSE	N
r_{10}	STU	CE	$ASGN$	CE	Y

4 Experimental Results

We evaluated the performance of HyPE on a number of synthetically generated data sets consisting of sets of subjects, objects, subject attribute-value matrices and object attribute-value matrices. The algorithm was implemented in Python 2.7.13 and executed on a 2.5 GHz Intel i5 CPU having 4 GB of RAM. We present the obtained results with number of subjects ($|S|$), number of objects ($|O|$), number of subject attributes ($|S_a|$), object attributes ($|O_a|$), subject attribute values ($|S_{av}|$), object attribute values ($|O_{av}|$), percentage of accesses resolved by the SO (N) and the execution time (T).

Table 3 shows the variation in the percentage of accesses resolved by the SO and the execution time with the number of subjects and the number of objects. It is seen that the percentage of accesses resolved by the SO doesn't change significantly when the number of subjects and objects is varied. We also see that the execution time of the algorithm increases with the number of subjects and object. Table 4 shows the variation in the percentage of accesses resolved by the SO and the execution time with the number of subject and object attributes. It is observed that the percentage of accesses resolved by the SO increases with the number of subject attributes and object attributes. When the number of subject and object attributes increases, it results in more unique attribute-value pair combinations which increases the percentage of accesses to be resolved by the SO. This also increases the execution time of HyPE.

Table 5 shows the variation in the percentage of accesses resolved by the SO and the execution time with the number of subject attribute values and object attribute values. It is seen that the percentage of accesses resolved by the SO increases with the number of subject and object attribute values. As the number of unique attribute-value pair combinations increases with the possible number of attribute values, it necessitates a greater percentage of accesses to be resolved by the SO. Similar to Table 4, less variation in execution time is observed. From Tables 4 and 5, it is seen that the number of possible values of an attribute has a greater influence on the percentage of accesses resolved by the SO than the number of subject and object attributes. Thus, HyPE performs well when many entities have similar attribute-value pairs and the possible number of values an attribute can have is small.

5 Related Work

Brickman et al. [3] present a reference design which uses commercially available technologies to demonstrate a sample ABAC platform. RBAC Policy-Enhance [1] provides a standard framework with specifications to handle the relationship between roles and dynamic constraints in RBAC, e.g., time of day,

TABLE 3: Variation in the percentage of accesses resolved by the SO and time (in sec.) for different $|S|$ and $|O|$

$\|S_{av}\| = 5, \|O_{av}\| = 5, \|S_a\| = 10, \|O_a\| = 10$								
$\|S\| = 100$			$\|S\| = 500$			$\|S\| = 1000$		
$\|O\|$	N	T	$\|O\|$	N	T	$\|O\|$	N	T
100	0.10	0.10	100	0.11	0.10	100	0.12	0.10
500	0.12	0.11	500	0.12	0.10	500	0.12	0.10
1000	0.12	0.11	1000	0.12	0.12	1000	0.12	0.12

TABLE 4: Variation in the percentage of accesses resolved by the SO and time (in sec.) for different $|S_a|$ and $|O_a|$

$\|S\| = 500, \|O\| = 500, \|S_{av}\| = 5, \|O_{av}\| = 5$								
$\|S_a\| = 5$			$\|S_a\| = 10$			$\|S_a\| = 20$		
$\|O_a\|$	N	T	$\|O_a\|$	N	T	$\|O_a\|$	N	T
5	0.06	0.07	5	0.11	0.12	5	0.13	0.16
10	0.10	0.11	10	0.14	0.13	10	0.19	0.18
20	0.13	0.15	20	0.18	0.17	20	0.22	0.22

TABLE 5: Variation in the percentage of accesses resolved by the SO and time (in sec.) for different $|S_{av}|$ and $|O_{av}|$

$\|S\| = 500, \|O\| = 500, \|S_a\| = 10, \|O_a\| = 10$								
$\|S_{av}\| = 5$			$\|S_{av}\| = 10$			$\|S_{av}\| = 20$		
$\|O_{av}\|$	N	T	$\|O_{av}\|$	N	T	$\|O_{av}\|$	N	T
5	0.11	0.10	5	0.28	0.35	5	0.70	0.62
10	0.28	0.34	10	0.63	0.59	10	1.44	0.83
20	0.73	0.65	20	1.39	0.81	20	2.38	1.04

location of access, etc. Coyne and Weil [4] discuss the possibility of simultaneously achieving the advantages of both RBAC and ABAC. Kuhn et al. [11] present a framework that merges RBAC and ABAC by adding attributes to RBAC. They claim that adding attributes to RBAC makes it suitable for distributed and dynamic environments. Bijon et al. [2] present a work on suitable attribute assignment to entities in an ABAC system to prevent unauthorized access. Xu and Stoller [19] propose a framework for mining parameterized role-based policies that supports a basic version of ABAC where roles are considered as subject attributes. Krautsevich et al. [10] construct an ABAC policy by defining values of attributes while deciding an access.

Vaidya et al. [18] present a change detection-based approach that enables policy migration. Given a set of policies of similar or distinct access control semantics, they find a common organizational policy with the lowest cost of migration. Xu et al. [20] propose the first known algorithm for mining ABAC policies using a bottom-up approach. Talukdar et al. [17] show that the problem of policy mining in ABAC is similar to that of identifying functional dependencies in database tables. In this context, they propose an ABAC policy mining algorithm which exhaustively enumerates all possible subject-object pairs. Das et al. [5] present a solution to the problem of policy mining in ABAC that uses Gini impurity to form an ABAC policy. They also include the environment attributes while mining the policy. Narouei et al. [14] propose a top-down policy engineering framework for ABAC that extracts policies from unrestricted natural language documents using a deep recurrent neural network.

All the above-mentioned approaches assist in ABAC policy engineering using either a top-down approach or a bottom-up approach. However, none of them use a hybrid approach for policy engineering as attempted by us.

6 Conclusion and Future Work

In this work, we have addressed the problem of policy engineering in ABAC using a hybrid approach. We have discussed a method which uses entropy and frequency to construct an ABAC policy. Future work in this domain includes the design of novel heuristics which are independent of parameters such as the number of attributes and the number of attribute values.

Acknowledgments

Research reported in this publication was supported by the National Institutes of Health under award R01GM118574 and by the National Science Foundation under awards CNS-1564034, CNS-1624503, and CNS-1747728. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

References

1. “Role based access control â€“ policy-enhanced,” *American National Standards Institute (ANSI)*, vol. INCITS 494-2012, 2012.
2. K. Z. Bijon, R. Krishnan, and R. Sandhu, “Towards an attribute based constraints specification language,” *International Conference on Social Computing (SocialCom)*, pp. 108–113, 2013.
3. N. Brickman, P. Burden, S. Jha, B. Johnson, A. Keller, T. Kolovos, S. Umarji, and S. Weeks, “Attribute based access control”. *NIST Special Publication*,” https://www.nccoe.nist.gov/projects/building-blocks/attribute-based-access-control'', 2017.
4. E. Coyne and T. R. Weil, “ABAC and RBAC: Scalable, flexible, and auditable access management,” *IT Professional*, pp. 14–16, 2013.
5. S. Das, S. Sural, J. Vaidya, and V. Atluri, “Using gini impurity to mine attribute-based access control policies with environment attributes,” *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 213–215, 2018.
6. D. Ferraiolo, R. Chandramouli, V. Hu, and R. Kuhn, “A comparison of attribute based access control (ABAC) standards for data service applications”. *NIST Special Publication*,” https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-178.pdf'', 2016.
7. M. R. Garey and D. S. Johnson, “Computers and intractability: A guide to the theory of np-completeness,” *W. H. Freeman & Co.*, 1979.
8. V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, “Guide to attribute based access control (ABAC) definition and considerations”. *NIST Special Publication*,” https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf'', 2014.
9. L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin, “Towards policy engineering for attribute-based access control,” *International Conference on Trusted Systems (INTRUST)*, pp. 85–102, 2013.
10. ——, “Towards attribute-based access control policy engineering using risk,” *International Workshop on Risk Assessment and Risk-driven Testing (RISK)*, pp. 80–90, 2014.
11. D. R. Kuhn, E. J. Coyne, and T. R. Weil, “Adding attributes to role-based access control,” *IEEE Computer*, pp. 79–81, 2010.
12. H. Lu, J. Vaidya, and V. Atluri, “An optimization framework for role mining,” *Journal of Computer Security*, vol. 22, no. 1, pp. 1–31, 2014. [Online]. Available: https://doi.org/10.3233/JCS-130484
13. B. Mitra, S. Sural, J. Vaidya, and V. Atluri, “A survey of role mining,” *ACM Computing Surveys*, pp. 50:1–50:37, 2016.
14. M. Narouei, H. Khanpour, H. Takabi, N. Parde, and R. Nielsen, “Towards a top-down policy engineering framework for attribute-based access control,” *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 103–114, 2017.
15. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *IEEE Computer*, pp. 38–47, 1996.
16. C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, pp. 379–423, 1948.
17. T. Talukdar, G. Batra, J. Vaidya, V. Atluri, and S. Sural, “Efficient bottom-up mining of attribute based access control policies,” *IEEE International Conference on Collaboration and Internet Computing (CIC)*, pp. 339–348, 2017.
18. J. Vaidya, B. Shafiq, V. Atluri, and D. Lorenzi, “A framework for policy similarity evaluation and migration based on change detection,” *International Conference on Network and System Security (NSS)*, pp. 191–205, 2015.
19. Z. Xu and S. D. Stoller, “Mining parameterized role-based policies,” *ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 255–266, 2013.
20. ——, “Mining attribute-based access control policies,” *IEEE Transactions on Dependable and Secure Computing (TDSC)*, pp. 533–545, 2015.