2017 IEEE 3rd International Conference on Collaboration and Internet Computing

Policy Adaptation in Attribute-Based Access Control for Inter-organizational Collaboration

Saptarshi Das[1], Shamik Sural[1], Jaideep Vaidya[2] and Vijayalakshmi Atluri[2]

***Abstract*— In Attribute-Based Access Control (ABAC), attributes are defined as characteristics of subjects, objects as well as environment, and access is granted or denied based on the values of these attributes. With increasing number of organizations showing interest in migrating to ABAC, it is imperative that algorithmic techniques be developed to facilitate the process. While the traditional ABAC policy mining approaches support the development of an ABAC policy from existing Discretionary Access Control (DAC) or Role-Based Access Control (RBAC) systems, they do not handle adaptation to the policy of a similar organization. As the policy itself need not be developed *ab initio* in this process, it provides agility and a faster migration path, especially for organizations participating in collaborative sharing of data. With the set of objects and their attributes given, along with an access control policy to be adapted to, the problem is to determine an optimal assignment of attributes to subjects so that a set of desired accesses can be granted. Here, optimality is in the number of ABAC rules the subjects would require to use to gain access to various objects. Such an approach not only helps in assisting collaboration between organizations, but also ensures efficient evaluation of rules during policy enforcement. We show that the optimal policy adaptation problem is NP-Complete and present a heuristic solution.**

I. INTRODUCTION

In any organization, it is imperative that valuable resources be protected from unauthorized accesses. Access control mechanisms are used to restrict access to various resources only to authorized entities. These protection requirements together specify the access control policy of the organization. Traditional access control policies are broadly based on two types of access control models, namely, Discretionary Access Control (DAC) [4] and Mandatory Access Control (MAC) [11]. In DAC, the owner of an object has the authority to grant or revoke access privileges to other subjects. The valid set of accesses is maintained in an Access Control Matrix (ACM) [12]. In contrast, MAC assigns a predefined security level to each subject and object, and is used primarily in government and military organizations. However, neither is used in most commercial organizations. Instead, Role-Based Access Control (RBAC) [13] is more commonly used, since it is policy neutral while being easier to manage. RBAC is based on the notion of *roles*, each of which is associated with a set of permissions. Users get their requisite permissions by being members of suitable roles. RBAC provides considerable flexibility in access control in situations involving a group of known users. But in dynamic and collaborative environments, the set of users may not be known *a priori*. This limitation makes RBAC unsuitable for use in organizations that require sharing of resources with others.

Attribute-Bases Access Control (ABAC) [5] has recently been proposed to enforce secure access to resources in a dynamic environment. This encourages, and often necessitates, organizations to migrate to ABAC from their traditional access control systems [14]. In ABAC, access decisions are based on the values of a set of *attributes*, each of which is a specific feature of a subject, object or environment, and a set of rules, collectively referred to as the organizational ABAC *policy*. Thus, an organization intending to migrate, needs an ABAC policy. Existing policy mining algorithms [18] facilitate the development of ABAC policies from DAC or RBAC systems. However, instead of developing a policy from scratch, re-using a standard policy or one that is being used by a similar organization helps in a faster migration process. Such a requirement quite often arises when any organization intends to collaborate with another for data sharing. The target organization, which is currently using a traditional access control model, therefore, needs to adapt itself to the existing ABAC policy of the collaborating organization through suitable assignment of attribute values to its entities, a process denoted as *policy adaptation* in this paper. Once the policy adaptation is complete, both the organizations will be using the same ABAC policy and depending on the attributes of users and objects, appropriate access will be given.

The main contribution of this work is that, we show a migration path in which an organization adapts to the ABAC policy of another organization with which it enters into a collaboration for data sharing and for a specific set of access requirements, determines an optimal assignment of attribute values to its subjects (interchangeably called users). In this context, optimality refers to the use of minimum number of rules by each subject to gain valid access on the objects after the migration process is complete and the ABAC system is put in place. This ensures efficient evaluation of rules and faster access decisions when an actual access request is made.

The rest of the paper is organized as follows. Section 2 discusses preliminaries and notations of ABAC. Section 3 gives a formal description of our proposed approach for policy adaptation in ABAC and studies its complexity. In Section 4, we describe a heuristic algorithm for optimal

[1]S. Das and S. Sural are with the Department of Computer Science and Engineering, IIT Kharagpur, India, saptarshidas13@iitkgp.ac.in, shamik@cse.iitkgp.ernet.in

[2]J. Vaidya and V. Atluri are with the MSIS Department, Rutgers Business School, USA, {jsvaidya,atluri}@rutgers.edu

0-7695-6303-1/17/31.00 ©2017 IEEE
DOI 10.1109/CIC.2017.00028

policy adaptation. Experimental results are presented in Section 5. Section 6 reviews related work. Finally, Section 7 concludes the paper and provides several directions for future research.

II. PRELIMINARIES

In this section, we introduce the terms and notations used for representing the various components of an ABAC system.

ABAC consists of a set of subjects, objects, environmental conditions and a set of access control rules. A subject usually denotes a human or an active non-person entity, such as an autonomous service or application. An object or resource is an entity that needs to be protected from unauthorized access. An environment defines the context in which an access request is made like *time of day*, *location of access*, etc. Every subject is associated with several attributes, such as *designation*, *experience*, etc., which either individually or in combination, comprises an expression to identify a group of subjects having similar access rights. Similarly, for each object, appropriate values are assigned to a set of object attributes. Typical examples of object attributes include *file type*, *sensitivity level* and *date of creation*. Rules define the access control policy of the organization. A set of formal notations is given below, which will be used throughout the paper.

- S: A set of authorized users. Each element of this set is represented as s_i, for $1 \leq i \leq |S|$.
- O: A set of objects which are to be protected. Each element of this set is represented as o_i, for $1 \leq i \leq |O|$.
- E: A set of environmental conditions. Each element of this set is represented as e_i, for $1 \leq i \leq |E|$.
- S_a: A set of subject attributes that can affect access decisions. Each element of this set is represented as sa_i, for $1 \leq i \leq |S_a|$. Each sa_i has a possible set of values it can acquire. For example, a subject attribute $designation$ may be associated with the set of possible values $\{professor, assistant\ professor, student, clerk, Null\}$. $Null$ indicates that the value of the attribute is not known for a subject. Similarly, O_a and E_a represent the sets of object attributes and environment attributes, respectively.
- F_s: $S \times S_a \rightarrow \{k|k$ is a subject attribute value$\}$. For example, consider a professor named Alice. Then, $F_s(Alice, designation) = professor$. The functions F_o and F_e are similarly defined for object and environment, respectively. Essentially, these functions assign values to attributes for all the entities.
- OP: A set of operations. Each element of this set is represented as op_i, for $1 \leq i \leq |OP|$. If $read$ and $modify$ are the only possible operations on objects, then $OP = \{read, modify\}$.
- R: A set of rules collectively called the ABAC policy. Each member of this set is represented as r_i, for $1 \leq i \leq |R|$.

Each rule $r \in R$ is a 4-tuple $< RS, RO, RE, op >$, where RS, RO and RE represent a set of subject attribute-value pairs, a set of object attribute-value pairs and a set of environment attribute-value pairs, respectively. op is the name of an operation. Each attribute-value pair $av \in \{RS \cup RO \cup RE\}$ is an equality of the form $a = c$, where a is the name of an attribute and c is the value associated with a. c is either a constant or a $don't\ care$ represented as "$-$". In this paper, we ignore the environment entity and its attributes for the sake of brevity. However, the work can be easily extended to handle environmental conditions also.

TABLE I: Subject attributes and their possible values

Subject attribute	Possible values of attribute
$designation$	$\{professor, assistantprofessor, student, clerk\}$
$department$	$\{CSE, EE\}$
$program$	$\{undergraduate, graduate\}$
$year\_of\_study$	$\{1, 2, 3, 4\}$

TABLE II: Object attributes and their possible values

Object attribute	Possible values of object attribute
$type$	$\{exam\ schedule, assignment, answer\ script\}$
$course$	$\{algorithms, circuit\ theory\}$

Example 1. Consider a university ABC that provides undergraduate and graduate courses in Computer Science and Engineering (CSE) and Electrical Engineering (EE). The subject, object and operation sets are given below:

- $S = \{Bob, Alice\}$
- $O = \{doc_1, doc_2, doc_3\}$
- $OP = \{read, evaluate\}$

Tables I and II, respectively represent the set of subject and object attributes along with their possible values. Tables III and IV give the values assigned to subject attributes and object attributes for each subject and object, respectively. Table V contains the ABAC rules of the university.

From Tables I-V, it can be concluded that, Bob is permitted to $read$ doc_1 and doc_2 using rules r_1 and r_2, respectively and $Alice$ gets the permission to $evaluate$ doc_3 using rule r_3.

The work presented in this paper enables an organization to adapt to an existing ABAC policy from a similar organization. In the following sections, we define the policy adaptation problem and propose a solution for it.

TABLE III: Subject to subject attribute value assignment

subject	designation	department	program	year_of_study
Bob	$student$	CSE	$undergraduate$	2
$Alice$	$professor$	EE	$Null$	$Null$

TABLE IV: Object to object attribute value assignment

O	type	course
doc_1	$exam\ schedule$	$Null$
doc_2	$assignment$	$algorithms$
doc_3	$answer\ script$	$circuit\ theory$

TABLE V: ABAC rules for ABC university

r_1	=<{*designation = -, department = CSE, s_course = undergraduate, year_of_study = 2*}, {*type = exam schedule, subject = -*}, *read*>
r_2	=<{*designation = student, department = CSE, s_course = undergraduate, year_of_study = 2*}, {*type = assignment, subject = algorithms*}, *read*>
r_3	=<{*designation = professor, department = EE, s_course = graduate, year_of_study = 1*}, {*type = answer script, subject = circuit theory*}, *evaluate*>

III. POLICY ADAPTATION PROBLEM (POLAP)

In this section, we first formally define the problem and then analyze its complexity.

A. Problem Definition

As mentioned in Section I, an organization desirous of migrating to ABAC by using a standard ABAC policy being used by another organization, must adapt itself through a suitable assignment of values to various attributes. Since object attributes typically represent inherent properties of the objects and they tend to be immutable, these are considered to be provided as input. The only features that can be algorithmically decided are the subject attributes. Keeping this in mind, we define Policy Adaptation Problem (PolAP) as the problem of assigning appropriate values to subject attributes with respect to an ABAC policy and a given set of desired accesses. While any valid assignment would have sufficed for this basic requirement, we pose an optimization version of PolAP that requires the assignments to be made in such a manner that the number of rules used by each subject for accessing the allowed objects is minimum. For notational simplicity, in the rest of the paper, we use PolAP to denote this optimization version. A formal definition of PolAP is given next.

Definition 3.1. PolAP
Given an ABAC policy R, a set S of subjects, set O of objects, set O_a of object attributes, set O_v of attribute-value pairs of all objects, set OP of possible operations and a desired set of accesses L, assign subject attribute-value pairs to all subjects such that all the desired accesses are satisfied and each $s_i \in S$ uses minimum number of rules R_i^{min} to gain all its accesses in L.

Thus, solving PolAP enables an organization to adapt to an existing ABAC policy through an optimal assignment of subject attribute-value pairs. An overarching requirement is that there must not be any security violation, i.e., it must be ensured that no extraneous access apart from the ones mentioned in L is introduced.

B. Complexity Analysis

In this sub-section, we do a formal analysis of the complexity of PolAP. We show that the problem is NP-Complete. To develop the proof, we first formulate a decision version of PolAP.

Definition 3.2. Decision Version of PolAP (D-PolAP)
Given R, L, S, O, O_a, O_v, OP as mentioned in Definition 3.1, and an integer k, does there exist a possible assignment of subject attribute-value pairs such that the required accesses in L are satisfied and the number of rules used by each $s_i \in S$ to obtain all its desired accesses is $\leq k$?

For proving NP-completeness of D-PolAP, we make use of a known NP-complete problem, namely, the Minimum Hitting Set (MHS) problem, which is defined below.

Definition 3.3. Minimum Hitting Set (MHS) Problem
Given a universe U and a collection ST of subsets of U, find the smallest subset $H \subseteq U$ such that every set in ST is hit by H, i.e., $\forall st_i \in ST : st_i \cap H \neq \phi$.

Definition 3.4. Decision Version of Minimum Hitting Set (D-MHS) Problem
Given a collection ST of subsets of a universal set U and an integer t, does there exist a hitting set H such that $|H| \leq t$.

Theorem 1. D-PolAP is NP-Complete.

Proof. We prove that D-PolAP is NP-Complete by first showing that D-PolAP is in NP, followed by reducing an instance of D-MHS [3] to an instance of D-PolAP in polynomial time.

Given a certificate consisting of a set of rules R, a collection R_i containing subsets of rules in R and a minimal set of rules R_i^{min}, it can be verified in linear time whether $|R_i^{min}| \leq k$ by counting the number of rules in R_i^{min}. It can also be verified that R and R_i are consistent with R_i^{min} by checking that the set of rules in R_i contains at least one rule from R_i^{min}. Thus, D-PolAP is in NP.

Now, we prove that D-MHS $\leq_p$ D-PolAP. Let $< U, ST, k >$ be an input instance of D-MHS. Then, an instance of D-PolAP can be generated in polynomial time by setting $R = U$ and $R_i = ST$. Here R is the set of all the rules s_i may use and R_i is a collection of subsets of R. Then, let H and R_i^{min} are solutions to the instances of D-MHS and D-PolAP, respectively. To complete the proof, we show that the output instance of D-MHS (consisting of H) is such that $|H|$ is less than or equal to k if and only if the output instance of D-PolAP (consisting of R_i^{min}) is such that $|R_i^{min}|$ has a value less than or equal to k.

Let H be the solution to a given instance of D-MHS such that $|H| \leq k$. Now, the solution to the instance of D-PolAP can be obtained from the solution to the given instance of D-MHS as follows: Since $U = R$, each element in U can be mapped to a rule in R. Similarly, each set of elements in ST can be mapped to a set of rules in R_i. Now, if H contains at least one element from each of the sets of elements in ST,

then R_i^{min} will also contain at least one rule from each of the sets of rules in R_i. Therefore, the solution of the instance of D-PolAP obtained from the solution of the instance of D-MHS is a valid solution. Similarly, it can be shown that a solution to D-MHS can be constructed from the solution to a given instance of D-PolAP. Hence, D-PolAP is NP-Hard.

Since D-PolAP is in NP and is NP-Hard, it is NP-Complete.

IV. HEURISTIC APPROACH FOR SOLVING POLAP

In this section, we present a heuristic approach for solving PolAP. The greedy heuristic used to solve the minimum hitting set problem [3] is applied to find a minimal set of rules for each subject in PolAP. The proposed heuristic algorithm is given in Algorithm 1.

Here, we first introduce an abstract data type called set-array (S-array), which is an array containing variable-size arrays of distinct elements. As the arrays in the S-array contain distinct elements, these are similar to mathematical sets. An S-array a containing n arrays of distinct elements is represented as $\Sigma_a[n]\{\ \}$. An example of S-array is:

$$\Sigma_a[3]\{\ \} = [\{a, b, c, d\}, \{a, e, f\}, \{u, v\}]$$

An element of an S-array can be accessed using its index, e.g., $\Sigma_a[1] = \{a, e, f\}$. $|\Sigma_a|$ denotes the number of arrays or sets in the S-array Σ_a. A set can be inserted at the end of an S-array by using the $insert(\Sigma_a, B)$ function, e.g., let $B = \{q, r, s\}$, then after performing $insert(\Sigma_a, B)$,

$$\Sigma_a = [\{a, b, c, d\}, \{a, e, f\}, \{u, v\}, \{q, r, s\}]$$

Algorithm 1 takes a set of objects O, a set of subjects S, an S-array of object attribute-value pairs Σ_o, a list of desired accesses L and a policy R as inputs, and for each subject, it assigns appropriate values to all the subject attributes and forms a set of rules using which the subject can satisfy all the required accesses in L. Algorithm 1 involves two phases, the first consisting of 4 steps and the second having 2 steps. In the following sub-sections we explain each phase in detail.

A. Rule Set Generation

For each subject, this phase constructs a collection of sets comprising rules. The rule set generation phase operates in four steps:

Step 1. Determine the set of rules affecting each object

This is the first step (Line 1) of Algorithm 1. For each object, it determines the set of rules using which that object can be accessed. The detailed procedure for creating such a set of rules for each object is given in Algorithm 2. A set of objects O, a policy R and an S-array of object attribute-value pairs Σ_o are given as inputs to Algorithm 2. For each object $o_i \in O$, the object attribute-value pairs associated with o_i are compared with the set of object attribute-value pairs of each of the rules in R. If the set of object attribute-value pairs associated with a rule $r_i \in R$ is also associated with o_i (Line 5), this indicates that the rule r_i can be used to access o_i. So, r_i is added to the set of rules for o_i (Line 6).

Algorithm 1: $PolAP(O, S, \Sigma_o, L, R)$

1. $\Sigma_a \leftarrow form\_rule\_set\_for\_objects(O, \Sigma_o, R)$
2. $\Sigma_b \leftarrow form\_object\_operation\_set\_for\_subjects$

   (S, L)
3. $\Sigma_c \leftarrow form\_object\_set\_for\_rules(O, \Sigma_o, R)$
4. $\Sigma_s \leftarrow [\ ]\{\ \}$
5. $\Sigma_r \leftarrow [\ ]\{\ \}$
6. **for** $i \leftarrow 1$ **to** $|S|$
7. $\quad \Sigma_u \leftarrow form\_rule\_set\_for\_subject$

   $\quad (i, \Sigma_o, \Sigma_a, \Sigma_b, \Sigma_c)$
8. $\quad$ **if** $\Sigma_U = NULL$
9. $\quad\quad$ **return** 0
10. $\quad\quad$ **exit**
11. $\quad H \leftarrow generate\_min\_hitting\_set(\Sigma_U)$
12. $\quad insert(\Sigma_r, H)$
13. $\quad$ **for** $j \leftarrow 1$ **to** $|H|$
14. $\quad\quad \Sigma_s[i] \leftarrow \Sigma_s[i] \cup \{H[j][i]\}$
15. **return** 1

Algorithm 2: $form\_rule\_set\_for\_objects(O, \Sigma_o, R)$

1. $\Sigma_a \leftarrow [\ ]\{\ \}$
2. **for** $k \leftarrow 1$ **to** $|O|$
3. $\quad C \leftarrow \{\ \}$
4. $\quad$ **for** $each\ r_i\ in\ R$
5. $\quad\quad$ **if** $r_i[RO] \subseteq \Sigma_o[k]$
6. $\quad\quad\quad C \leftarrow C \cup \{r_i\}$
7. $\quad insert(\Sigma_a, C)$
8. **return** Σ_a

Step 2. Determine the set of object-operation pairs for each subject

The second step of Algorithm 1 (Line 2) iterates over all the 3-tuples of the list of accesses L and forms the object-operation pairs which are permissible to each subject. The detailed steps for generating such allowable object-operation pairs is given in Algorithm 3. Algorithm 3 takes a set of subjects S and a list of desired accesses L as inputs. Each element of the list of accesses is represented as a 3-tuple $< s_i, o_j, op_k >$. This implies that subject s_i is permitted to perform operation op_k on object o_j. A set of object-operation pairs is constructed for each $s_i \in S$. Then each element $l_i \in L$ is evaluated and the object and operation in l_i are added to the set of object-operation pairs for the subject in l_i. For example, when the 3-tuple $< s_i, o_j, op_k >$ is evaluated,

Algorithm 3:
$form\_object\_operation\_set\_for\_subjects(S, L)$

1. $\Sigma_b \leftarrow [|S|]\{\ \}$
2. **for** $each\ (s_i, o_j, op_k)\ in\ L$
3. $\quad \Sigma_b[i] \leftarrow \Sigma_b[i] \cup \{(o_j, op_k)\}$
4. **return** Σ_b

Algorithm 4: $form\_object\_set\_for\_rules(O, \Sigma_o, R)$

1. $\Sigma_c \leftarrow [\,]\{\,\}$
2. **for** *each* r_i *in* R
3. $\quad C \leftarrow \{\,\}$
4. $\quad$ **for** $k \leftarrow 1$ **to** $|\Sigma_o|$
5. $\quad\quad$ **if** $r_i[RO] \subseteq \Sigma_o[k]$
6. $\quad\quad\quad C \leftarrow C \cup \{O[j]\}$
7. $\quad insert(\Sigma_c, C)$
8. **return** Σ_c

(o_j, op_k) is added to the set of object-operation pairs for s_i. Effectively, for each subject, a set is generated which contains the object-operation pairs permitted to the subject according to the list of accesses L.

Step 3. Determine the set of objects covered by each rule The third step (Line 3) of Algorithm 1 is the dual of the first step, i.e., it iterates over all the rules and for each rule, it generates the set of objects which can be accessed using that rule. The detailed steps are given in Algorithm 4. A set of rules R, a set of objects O and an S-array of object attribute-value pairs Σ_o are given as its inputs.

For each rule $r_i \in R$, the object attribute-value pairs associated with r_i are compared with the set of object attribute-value pairs associated with each object. If the object attribute-value pairs associated with a rule $r_i \in R$ are also associated with o_i (Line 5), this indicates that object o_i can be accessed through rule r_i. So, o_i is added to the set of objects for r_i (Line 6). Essentially, for each rule, a set of objects is generated such that, each object in the set can be accessed through that rule.

Step 4. Generate collection of rule sets for each subject The fourth step (Lines 6-10) of Algorithm 1 generates a collection of sets of rules, where each set consists of rules that can be used to access a specific object-operation pair allowed to a subject. Therefore, for a subject, the collection of sets contains a set of rules corresponding to each object-operation pair permitted to the subject. The collection of rule sets for subjects is generated by constructing a Subject-Object-Rule-Object (SORO) tree for each subject. The detailed steps of the algorithm for building a SORO tree is given in Algorithm 5.

Definition 4.1. Subject-Object-Rule-Object (SORO) tree *A* SORO *tree is an n-ary tree consisting of* 4 *levels. The root (first level) of a* SORO *tree represents a subject. The second level consists of the object-operation pairs permitted to that subject. The third level is built using the rules through which the objects in the previous level can be accessed. The fourth level of the tree consists of the objects accessible through the rules in the previous level.*

The SORO tree for a given subject $s_i \in S$ is constructed by first initializing the tree with s_i as the root. All the object-operation pairs permitted to s_i are added as child nodes of s_i. These object-operation pairs are obtained from

Algorithm 5: $form\_rule\_set\_for\_subject(i, \Sigma_o, \Sigma_a, \Sigma_b, \Sigma_c)$

1. $S \leftarrow \Sigma_b[i]$
2. $S' \leftarrow \{\,\}$
3. **for** *each* (o_s, op_t) *in* S
4. $\quad S' \leftarrow S' \cup \{o_s\}$
5. $\Sigma_r \leftarrow \{\,\}$
6. **for** *each* (o_m, op_k) *in* S
7. $\quad T \leftarrow \Sigma_a[m]$
8. $\quad$ **for** *each* r_d *in* T
9. $\quad\quad$ **if** $r_d[op] \neq op_k$
10. $\quad\quad\quad \Sigma_a[m] \leftarrow \Sigma_a[m] - \{r_d\}$
11. $\quad insert(\Sigma_r, \Sigma_a[m])$
12. $\Sigma_r' \leftarrow \Sigma_r$
13. **for** $j \leftarrow 1$ **to** $|\Sigma_r'|$
14. $\quad$ **for** *each* r_q *in* $\Sigma_r'[j]$
15. $\quad\quad R \leftarrow \Sigma_c[q]$
16. $\quad\quad$ **if** $(R - S') \neq \{\,\}$
17. $\quad\quad\quad \Sigma_r[j] \leftarrow \Sigma_r[j] - \{r_q\}$
18. $\quad\quad\quad$ **if** $\Sigma_r[j] = \{\,\}$
19. $\quad\quad\quad\quad$ **return** 0
20. $\quad\quad\quad\quad$ **exit**
21. **return** Σ_r

the sets of object-operation pairs generated by Algorithm 3. This completes the second level of the SORO tree. For each object-operation pair in the second level, the rules through which they can be accessed are added as child nodes of the object-operation pair to complete the third level of the SORO tree. The rules which allow access to a specific object-operation pair are obtained from the sets of rules generated by Algorithm 2. Finally, for each rule in the third level, objects that can be accessed using that rule are added as child nodes of the rule. The objects accessible through a given rule are obtained from the sets of objects generated by Algorithm 4. Construction of the SORO tree is followed by the comparison of the objects in the leaf nodes of the SORO tree with the objects present in the second level. If any extra object is found in the leaf nodes, the parent rule of that extraneous object in the third level is removed from the SORO tree along with all its children. An extraneous object in the leaf node represents an unauthorized access. Removal of rules allowing unauthorized accesses is called *pruning*. After pruning, the rules in the third level of the SORO tree are grouped into sets with respect to the object-operation pairs present in the second level.

This marks the end of the first phase of Algorithm 1. At this stage, for each subject, we have a collection of sets of rules.

B. Generation of Minimal Rule Set and Subject Attribute-Value Pairs

The second phase works in two steps. From the collection of sets of rules obtained at the end of the first phase, for each subject, this phase first creates a minimal set of rules using which the subject can satisfy its required accesses, and then assigns attribute-value pairs to the subjects.

Algorithm 6: $generate\_min\_hitting\_set(\Sigma_s)$

1. $U \leftarrow \{\ \}$
2. **for** $i \leftarrow 1$ **to** $|\Sigma_s|$
3. $\quad U \leftarrow U \cup \{\Sigma_s[i]\}$
4. $S^{'} \leftarrow \{\ \}$
5. **while** $S \neq NULL$
6. $\quad select\ u_i\ from\ U\ s.t.\ u_i\ hits\ most\ member\ of\ sets\ from\ \Sigma_s$
7. $\quad remove\ already\ hit\ sets\ from\ \Sigma_s$
8. $\quad S^{'} \leftarrow S^{'} \cup \{u_i\}$
9. **return** $S^{'}$

Step 5. Determine Minimal Rule Set for Subjects
The fifth step (Line 11) of Algorithm 1 computes a minimal set of rules required by each subject to satisfy all the desired accesses. It takes the rule sets generated by the previous step and applies the $generate\_min\_hitting\_set$ procedure given in Algorithm 6. The minimal set of rules is computed using a greedy heuristic, where in each step, the rule which is a member of the maximum number of rule sets is selected. In situations where there are more than one minimum hitting set, any one may be randomly selected. This procedure is similar to finding a minimum hitting set from a collection of subsets of a universe. Here, in each step, the element occurring in the maximum number of un-hit sets is selected.

Step 6. Subject Attribute-Value Pairs Generation
The sixth and final step (Lines 13-14) of Algorithm 1 takes the set of rules generated in Step 5 as input. For each subject, the subject attribute-value pairs associated with each rule in the minimal set of rules for the subject are assigned to that subject. At the end of the second phase of Algorithm 1, for each subject, we obtain a set of rules using which it can satisfy all its desired accesses. Also, we get the subject attribute-value pairs assigned to all the subjects.

Illustrative Example

To illustrate the working principle of our approach we consider that a university XYZ plans to migrate to ABAC. In the process, it would like to reuse the ABAC policy given in Table VI. The object attributes, their possible values as well as object to object attribute value assignments for XYZ are given in Tables II and IV, respectively.

$Cathy$ is an employee of university XYZ who should have the permissions to $read\ doc_1$ and $evaluate\ doc_3$. Now, we show how university XYZ adapts to the desired ABAC policy by assigning the subject attribute-value pairs to $Cathy$.
Algorithm 2 constructs a set of rules for each object, i.e., doc_1, doc_2 and doc_3 by evaluating the object attribute-value pairs associated with each object against the set of object attribute-value pairs in each rule. The objects and the rules through which they can be accessed are given in Table VII. Algorithm 3 constructs the set of allowed object-operation pairs for $Cathy$. Table VIII gives the object-operation pairs permitted to $Cathy$.

Algorithm 4 constructs a set of objects for rules r_1, r_2 and r_3 by comparing the set of object attribute-value pairs in each rule with the object attribute-value pairs associated with each object. The rules and the objects that are accessible through them are given in Table IX.

Algorithm 5 generates the sets of rules for $Cathy$ corresponding to the object-operation pairs given in Table VIII by generating a SORO tree as described in Section 4.1. The SORO tree for $Cathy$ is given in Figure 1. Initially, the sets of rules generated are $\{r_1, r_2\}$ and $\{r_3\}$. According to Table IX, rule r_1 allows $Cathy$ to access doc_2. As per Table VIII, accessing doc_2 is not permitted to $Cathy$. So, r_1 is removed from the sets of initially generated rules which finally becomes $\{r_2\}$ and $\{r_3\}$. The pruned SORO tree for $Cathy$ is given in Figure 2.

Algorithm 6 computes a minimal set of rules for $Cathy$ through which she can obtain all the desired accesses. The minimal set of rules for $Cathy$ is $\{r_2, r_3\}$. The set of subject attribute-value pairs associated with the rules r_2 and r_3 are assigned to $Cathy$. The subject attribute value assignments for $Cathy$ are shown in Table X.

V. Experimental Results

We tested the performance of the proposed approach for policy adaptation with the data sets given in [18]. Further, we evaluated the algorithms on a number of synthetically generated ABAC policies, lists of accesses, sets of subjects, objects, operations and object to object attribute value assignments. We implemented the algorithm in Python 3.4.4 and executed on a 3.2 GHz Intel i5 CPU having 4 GB of RAM. We present the obtained results using number of subjects ($|S|$), number of subject attributes ($|S_a|$), number of objects ($|O|$), number of object attributes ($|O_a|$), number of rules ($|R|$), average size of the set of rules used by a subject ($|R_s|$) and the time taken T for executing the algorithms in seconds.

Table XI shows the average size of the set of rules used by a subject and time taken for the execution of the proposed algorithm for policy adaptation on the various data sets given in [18]. It is observed that, as the number of subjects and objects increases, the average number of rules used by each subject also increases. This can be attributed to the increase in the number of objects. Subjects are more likely to access higher number of objects if the number of objects is increased, which in turn, increases the number of rules required to fulfill the accesses. The execution time of the algorithm also increases with increasing number of subjects as it takes one iteration of the algorithm for generating a minimal set of rules for one subject.

Next, we evaluate the proposed algorithm on various synthetically generated data sets. When the number of subjects, objects and object attributes are kept constant, we consider two cases that may arise: (i) variation in the number rules while keeping the number of subject attributes fixed (ii) variation in the number of subject attributes keeping the number of rules fixed. Similarly, two cases arise when the number of subjects, objects and subject attributes are kept constant. Further, we show the effect of varying the number

TABLE VI: ABAC policy to be re-used

r_1	=<{*age = -, designation = -, department = -, s_course = undergraduate, year_of_study = 2*}, {*type = -, subject = -*}, *read*>
r_2	=<{*age = -, designation = -, department = EE, s_course = -, year_of_study = -*}, {*type = exam schedule, subject = -*}, *read*>
r_3	=<{*age = -, designation = professor, department = -, s_course = -, year_of_study = -*}, {*type = answer script, subject = circuit theory*}, *evaluate*>

Object	Rule(s)
doc_1	r_1, r_2
doc_2	r_1
doc_3	r_3

TABLE VII: Objects and rules through which they can be accessed

Rule	Object(s)
r_1	doc_1, doc_2
r_2	doc_1
r_3	doc_3

TABLE IX: Objects accessible through each rule

Subject	Object-operation pairs
$Cathy$	$(doc_1, read), (doc_3, evaluate)$

TABLE VIII: Object-operation pairs permitted to a subject

subject	designation	department
$Cathy$	$professor$	EE

TABLE X: Subject to subject attribute value assignment for $Cathy$

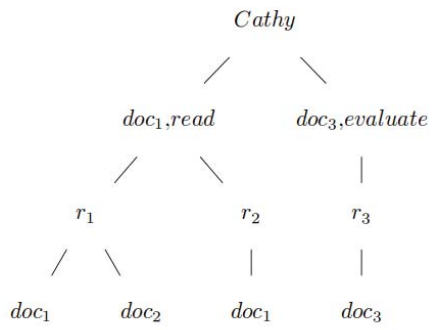


Fig. 1: SORO tree for $Cathy$

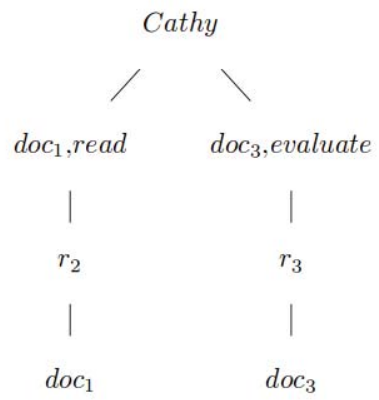


Fig. 2: SORO tree for $Cathy$ after pruning rule r_1

of subjects and objects on the average number of rules used by each subject.

Table XII shows the variation in the average number of rules used by a subject and the execution time with the number of rules and the number of subject attributes. It is seen that the impact of variation in the number of subject attributes on the average number of rules used by each subject is lesser. Also, the number of rules used by a subject increases with increase in the total number of rules. For 20, 30, 40 and 50 rules, the average percentage of number of rules to be evaluated for deciding an access request is found to be 17.85%, 15.3%, 13.2% and 12.18%, respectively (computed by taking the maximum value of $|R_s|$ for a given $|R|$). This implies faster evaluation of rules and hence efficient access decisions. The percentage of rules to be evaluated decreases with increasing number of rules. Hence, adapting to larger policies is more beneficial. We also see that, the execution time of the algorithm increases with increasing number of rules while relatively less variation is seen with increase in the number of subject attributes.

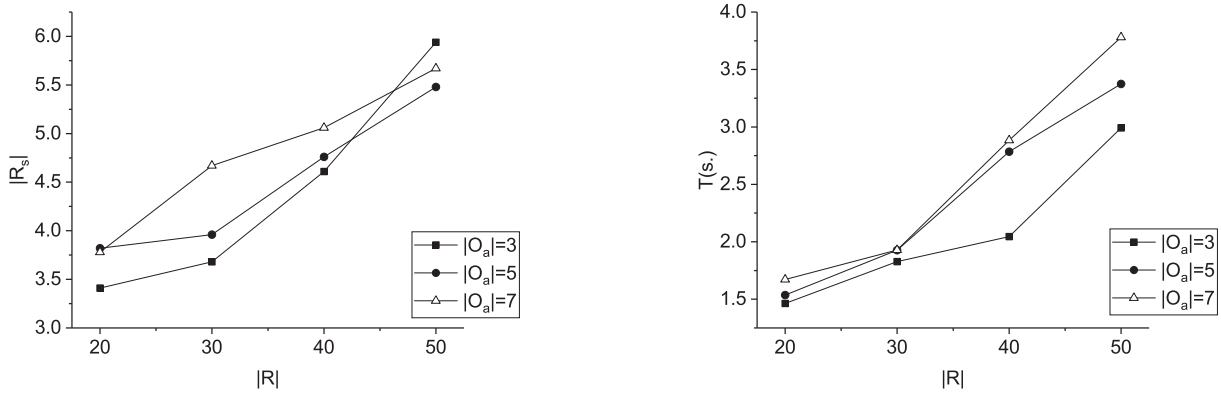
Figure 3a shows the variation in the average number of rules used by a subject (y-axis) with number of rules (x-axis) for various number of object attributes. Similar to the results given in Table XII, here also it is seen that, variation in the average number of rules used by each subject when the number of object attributes is varied, is less and the number of rules used by a subject increases with increase in the total number of rules. For 20, 30, 40 and 50 rules, the average percentage of number of rules to be evaluated for deciding an access request is found to be 19.1%, 15.56%, 12.65% and 11.8%, respectively (computed by taking the maximum value of $|R_s|$ for a given $|R|$). Here also the percentage of rules to be evaluated decreases with increasing number of rules. Figure 3b shows the variation in execution time of the

TABLE XI: Variation of average size of rule set and time for the data sets in [18]

Dataset	$\|S\|$	$\|S_a\|$	$\|O\|$	$\|O_a\|$	$\|R\|$	$\|R_s\|$	T(in sec.)
Online video library	12	2	13	2	6	1.23	0.01
Healthcare	21	6	16	7	11	1.33	0.02
University	20	6	34	5	10	1.40	0.02
Project management	16	7	40	6	19	1.25	0.03

TABLE XII: Variation of average size of rule set and time for different number of subject attributes and objects

$\|S\| = 100$, $\|O\| = 200$, $\|O_a\| = 5$											
$\|R\| = 20$			$\|R\| = 30$			$\|R\| = 40$			$\|R\| = 50$		
$\|S_a\|$	$\|R_s\|$	T(in s.)	$\|S_a\|$	$\|R_s\|$	T(in s.)	$\|S_a\|$	$\|R_s\|$	T(in s.)	$\|S_a\|$	$\|R_s\|$	T(in s.)
3	3.57	1.21	3	4.48	2.02	3	4.89	2.17	3	6.09	2.78
5	3.24	1.02	5	4.07	1.98	5	5.28	2.52	5	5.42	4.01
7	3.44	0.97	7	4.59	1.98	7	4.97	2.68	7	5.61	3.84



(a) Variation in the average number of rules used by a subject

(b) Variation in execution time

Fig. 3: Performance with varying number of rules

algorithm (y-axis) with the total number of rules (x-axis). It increases with the increase in the number of rules and relatively less increase is observed when the number of object attributes is increased.

The results given so far are on a fixed number of subjects and objects. Table XIII shows the variation of the average number of rules used by each subject and the time taken for policy adaption with variation in the number of subjects and objects. This table shows that the average number of rules used by each subject increases with increase in the number of objects.

Finally, Figure 4 shows the variation in the average number of rules used by each subject (y-axis) with number of subjects (x-axis) for different number of objects. From the figure, it is seen that the average number of rules used by a subject increases with increase in the number of objects.

VI. Related Work

Existing approaches for security policy migration includes algorithms that use attribute data. Colantonio *et al.* [2] use attribute data and an access control list for mining role-based policies. Molloy *et al.* [9] and Xu and Stoller [17] form RBAC policies with low weighted structural complexity (WSC). These policies consist of meaningful roles such that the user to role mapping can be expressed as a predicate over a set of subject attribute-value pairs. One major limitation of these algorithms is that they use attributes like *uid* which uniquely identifies a user in a user to role assignment. This results in some users not being characterized by a subject attribute-value pair expression.

Xu and Stoller [16] proposed a parameterized RBAC (PRBAC) framework for mining parameterized role-based policies that supports a basic version of ABAC because roles are characterized by user attributes. Ni *et al.* [10] utilize machine learning algorithms for security policy mining. They use supervised learning techniques to train classifiers for associating permissions and roles. Their algorithm takes permissions, roles and role to permission assignments as training data. The output is a support vector machine (SVM) classifier that automatically allocates new permissions to roles. In addition to that, they also consider another scenario where classifiers are trained using a supervised learning algorithm to automate user to role assignments. The users, roles, user attribute values and user to role assignments are given as input to the classifier. Lim *et al.* [8] utilize

TABLE XIII: Variation of average size of rule set and time for different number of subjects and objects

$\|S_a\| = 5$, $\|O_a\| = 5$, $\|R\| = 20$														
$\|S\| = 10$			$\|S\| = 20$			$\|S\| = 30$			$\|S\| = 40$			$\|S\| = 50$		
$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)
30	2.64	0.02	30	2.83	0.02	30	2.47	0.02	30	3.33	0.04	30	2.72	0.05
50	3.13	0.03	50	3.08	0.04	50	3.13	0.04	50	3.07	0.06	50	3.02	0.09
100	3.56	0.05	100	3.92	0.08	100	3.31	0.09	100	3.48	0.12	100	3.58	0.19
$\|S_a\| = 5$, $\|O_a\| = 5$, $\|R\| = 20$														
$\|S\| = 60$			$\|S\| = 70$			$\|S\| = 80$			$\|S\| = 90$			$\|S\| = 100$		
$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)	$\|O\|$	$\|R_s\|$	T(in s.)
30	2.26	0.04	30	2.81	0.07	30	3.33	0.09	30	3.13	0.11	30	3.34	0.11
50	3.15	0.07	50	3.72	0.17	50	3.82	0.12	50	3.24	0.15	50	3.28	0.16
100	3.49	0.15	100	3.82	0.19	100	3.63	0.25	100	3.89	0.34	100	3.93	0.37

evolutionary algorithms to construct and improve security policies. Though they do not consider ABAC policy mining, some complex problems involving risk-based policies are surveyed by them. Agrawal *et al.* [1] present an algorithm for mining propositional rules. The limitation of their approach is that they are unable to construct a set of rules which are exactly satisfied. The probabilistic nature of the generated rules makes it unsuitable for ABAC policy mining. Apart from this, the set of generated rules fails to cover all the input data.

Vaidya *et al.* [15] present a framework for migrating to ABAC. Their work is based on a change detection approach that is used to evaluate similarities between security policies of similar or distinct access control semantics. Given a set of policies, they find a common organizational policy with the lowest cost of migration. Xu *et al.* [18] proposed the first known algorithm for mining ABAC policies from access control lists and attribute data. They also provide an extension of the algorithm to detect over-assignment and under-assignment of permissions to a user.

John *et al.* [6] present a framework for identification of a correct set of ABAC rules for establishing secure collaborations among multiple clouds. In this paper, they formally define cross-domain rule mining as the problem of finding a minimal set of ABAC rules that allow access to the resources of one cloud by the users of another cloud. In another work John *et al.* [7] propose 2 approaches for inter-cloud rule formation in ABAC. In the first approach, they consider cross domain rule mining as the problem of forming a minimal set of positive authorizations only. The second approach shows the advantage of developing deny rules along with positive authorizations in reducing the total number of rules, and hence, the response time for evaluating access requests. All the above-mentioned approaches assist in security policy mining. However, none of them use policy adaptation as done by us, which is required for setting up collaboration between organizations with one using ABAC and the other using a traditional access control model. To the best of our knowledge, this is the first ever approach towards optimal policy adaptation for migrating to ABAC.

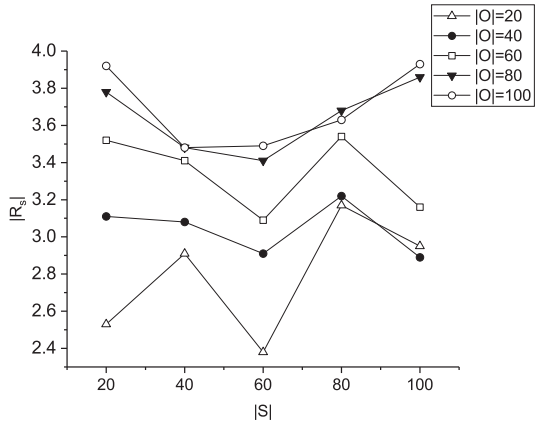


Fig. 4: Variation of the average number of rules used by each subject with the number of subjects

VII. Conclusion and Future Work

In this paper, we have introduced the problem of policy adaptation, which needs to be solved for an organization for migrating to ABAC using an existing policy. The optimal policy adaptation problem has been shown to be NP-Complete and a heuristic solution has been proposed. Future work in this area would include designing of novel heuristics that can further reduce the number of rules used by each subject. Other optimization metrics besides the number of rules may also be defined that could generate more meaningful subject attribute-value pairs.

References

[1] Agrawal, Rakesh and Srikant, Ramakrishnan: Fast algorithms for mining association rules. International Conference on VLDB, 487–499 (1994)

[2] Colantonio, Alessandro and Di Pietro, Roberto and Verde, Nino Vincenzo: A Business-driven Decomposition Methodology for Role Mining. Computer Security, 844–855 (2012)

[3] Gainer-Dewar, Andrew and Vera-Licona, Paola: The minimal hitting set generation problem. Algorithms and computation, (2016)

[4] Harrison, Michael A. and Ruzzo, Walter L. and Ullman, Jeffrey D.: Protection in Operating Systems. Commun. ACM, 461–471 (1976)

[5] Hu, Vincent C and Ferraiolo, David and Kuhn, Rick and Schnitzer, Adam and Sandlin, Kenneth and Miller, Robert and Scarfone, Karen: Guide to attribute based access control (ABAC) definition and considerations. National Institute of Standards and Technology Special Publication, (2012)

[6] John, John C. and Sural, Shamik and Gupta, Arobinda: Authorization Management in Multi-cloud Collaboration Using Attribute-Based Access Control. International Symposium on Parallel and Distributed Computing (ISPDC), 190–195, (2016)

[7] John, John C. and Sural, Shamik and Gupta, Arobinda: Attribute-based access control management for multicloud collaboration. Concurrency and Computation: Practice and Experience, (2017)
[8] Lim, Y.T: Evolving security policies. Ph.D. dissertation, University of York, (2010)
[9] Molloy, Ian and Chen, Hong and Li, Tiancheng and Wang, Qihua and Li, Ninghui and Bertino, Elisa and Calo, Seraphin and Lobo, Jorge: Mining Roles with Multiple Objectives. ACM Transactions on Information and System Security (TISSEC), 36:1–36:55 (2010)
[10] Ni, Qun and Lobo, Jorge and Calo, Seraphin and Rohatgi, Pankaj and Bertino, Elisa: A Business-driven Decomposition Methodology for Role Mining. ACM Symposium on Access Control Models and Technologies (SACMAT), 75–84 (2009)
[11] Sandhu, Ravi S.: Lattice-Based Access Control Models. Computer, 9–19 (1993)
[12] Sandhu, Ravi S., and Pierangela Samarati: Access control: principle and practice. IEEE communications magazine 32, (1994)
[13] Sandhu, Ravi S. and Coyne, Edward J. and Feinstein, Hal L. and Youman, Charles E.: Role-Based Access Control Models. IEEE Computer, 38–47 (1996)
[14] Servos, Daniel and Osborn, Sylvia L.: Current Research and Open Problems in Attribute-Based Access Control. ACM Computing Surveys, 65:1–65:45 (2017)
[15] Vaidya, Jaideep and Shafiq, Basit and Atluri, Vijayalakshmi and Lorenzi, David: A Framework for Policy Similarity Evaluation and Migration Based on Change Detection. Network and System Security: 9th International Conference, 191–205 (2015)
[16] Xu, Zhongyuan and Stoller, Scott D.: Mining Parameterized Role-based Policies. ACM Conference on Data and Application Security and Privacy (CODASPY), 255–266 (2013)
[17] Xu, Zhongyuan and Stoller, Scott D.: Algorithms for Mining Meaningful Roles. ACM Symposium on Access Control Models and Technologies (SACMAT), 57–66 (2012)
[18] Xu, Zhongyuan and Stoller, Scott D.: Mining Attribute-Based Access Control Policies. IEEE Transactions on Dependable and Secure Computing, 533–545 (2015)