EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Supervised Learning-Based Approach Mining ABAC Rules from Existing RBAC Enabled Systems

Gurucharansingh Sahani[1,*], Dr. Chirag S. Thaker[2], and Dr. Sanjay M. Shah[3]

[1]Ph.D. Scholar Computer/IT, Gujarat Technological University, Gandhinagar, India
[2]Professor, LD College of Engineering, Ahmedabad, India
[3]Professor, Government College of Engineering, Rajkot, India

Abstract

Attribute-Based Access Control (ABAC) is an emerging access control model. It is the more flexible, scalable, and most suitable access control model for today's large-scale, distributed, and open application environments. It has become an emerging research area nowadays. However, Role-Based Access Control (RBAC) has been the most widely used and general access control model so far. It is simple in administration and policy definition. But user-to-role assignment process of RBAC makes it non-scalable for large-scale organizations with a large number of users. To scale up the growing organization, RBAC needs to be transformed into ABAC. Transforming existing RBAC systems into ABAC is complicated and time-consuming. In this paper, we present a supervised machine learning-based approach to extract attribute-based conditions from the existing RBAC system to construct ABAC rules at the primary level and simplify the process of the transforming RBAC system to ABAC.

Keywords: Attribute-based Access Control (ABAC), Role-Based Access Control (RBAC), Mining ABAC Rule, Supervised Machine Learning

Received on 22 June 2022, accepted on 04 September 2022, published on 07 September 2022

Copyright © 2022 Gurucharansingh Sahani *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.v5i16.1560

*Corresponding author. Email: gurcharan_sahani@yahoo.com

1. Introduction

The access control model defines how the access of system resources to users is controlled and how information about who is authorized for what is maintained. An access control policy is how the system knows who is authorized for what operations on system resources. Traditional access control models Discretionary Access Control (DAC) [1, 2, 3, 4], Mandatory Access Control Model (MAC) [4, 5, 6] and RBAC [2, 7, 8, 9] are applicable in several application scenarios. However, they have their limitations and disadvantages in the current large, dynamic, and distributed application domains [10, 11, 12]. They do not satisfy the need of growing organizations that span over large geographical areas.

ABAC being flexible and context-sensitive satisfies today's application environments [13, 14, 15, 16]. It is based on the concept of the user, object, and environmental attributes of the system. Access to the system resources is controlled by the access rules defined over these attributes. These rules are the set of conditional expressions over the user, object and environmental attributes joined with "and" connectives. For example designation, age, experience, etc. can be the user attributes, department of an object can be object attribute while time and location of the user can be the environmental attributes. The user, object, and environment attribute values at the time of request need to satisfy the conditions defined in the rule to acquire particular permission. For example, a policy rule can be stated as: *"A permission to evaluate the grade-sheet of the student is granted to the user if and only if his Designation is Assistant Professor, Age >= 25 years,*

Experience >=3 years, his Department = CS and the time of the request is in between 10 am to 5 pm". Defining ABAC policy is a complex and time-consuming process. Constructing ABAC policy for newly designed applications is worthy. However, it is better if we can reduce the effort of constructing them for an existing application that has implemented RBAC as an access control model.

An earlier number of approaches [17, 18, 19, 20, 21, 22] on role mining have been proposed. However, they do not address the problem of ABAC policy mining. Recently there has been little research carried out on the automatic mining of ABAC policies. These approaches utilize access control logs and the Access Control List (ACL) of the system to mine the ABAC rules. Techniques they have used are either association mining or unsupervised machine learning. Association mining can extract the patterns of the attribute values from the log entries and provide association among them. They do not extract conditional expressions with inequalities as shown in the above example. The unsupervised machine learning approach has issues related to clustering algorithms. It also results in mining a large number of roles. All of these approaches work better for categorical data. But, do applicable for discrete data.

In this paper, we propose a supervised machine learning approach mainly for the applications that need to transform the access control model from RBAC to ABAC.

The paper is organized as follows. Section 2 describes related work. Section 3 gives background knowledge and preliminaries for the RBAC and ABAC models. Section 4 explains our approach. Section 5 contains results and discussion. Section 6 concludes the paper.

2. Related Work

Recently little bit of research has been carried out on mining ABAC rules [19, 23, 24, 25, 26, 27]. Amani et al [28] have proposed an automatic approach to extract ABAC rules from event logs of business processes. They have used association rule mining to find associations of the subject, object, and environmental attributes. Association mining is applied on the event log to identify attribute relations using frequent and in-frequent item sets. ABAC rules are then constructed by using attribute relations.

Zhongyuan and Scott [27] have presented an ABAC policy mining algorithm using Access Control Lists (ACLs) and attribute data. The access control list is a set of user-permission relations as tuple $< u, o, op >$ containing user, object, and operation. The algorithm defined by the authors iterates over such tuples selects some tuples as seeds to construct candidate ABAC rules and attempts to generalize the candidate rules by considering additional tuples in the ACL. The generated rules are merged, and simplified and highly-quality rules are finally considered for constructing the ABAC policy.

Carlos et al propose [29] the model to extract ABAC rules from sparse logs. This model is also based on association rule mining. The authors have highlighted the fact that most of the logs in practice contain less amount of information. The model invalidates overly permissive rules and reduces the excessive rules generated by the mining process. Overly permissive rules are the rules such assign permission to users that are undesirable according to security perspectives. Standard association mining algorithms fail to identify these rules. For this purpose, a new quality measure has been defined to guide the mining process.

An automated ABAC rule mining proposed by Matthew W Sanders and Chuan Yue [24] addresses the under-privilege and over-privileged issues related to access control. Over-privilege increases the risk of security in the system while under-privilege restricts users from performing their duties. The rule mining algorithm they have presented minimized the privilege errors of ABAC policies. They have presented the new algorithms, evaluation metrics, and optimization methods to optimize the large privilege space of ABAC policy.

Karimi et al. [23] proposed a methodology for extracting ABAC policy rules. They have used an unsupervised learning approach for mining policy rules from the system log. Transaction records of the system log are grouped into clusters. Each cluster corresponds to one access control rule. Features of the cluster are used to construct policy rules. As the approach is using a clustering algorithm, it has all the issues related to the clustering algorithm. Similarly, the approach fails to extract attribute conditions for non-categorical attributes.

Most of the approaches are based on association rule mining leading to the large number of rule extraction needing pruning methods to eliminate irrelevant rules. An unsupervised learning-based approach also results in inaccurate access rules due to the issues related to clustering algorithms. All of these approaches do not able to extract attribute conditional relations for discrete data attributes. For example, they can extract the conditions like $\{dept = CS\}$ for categorical attributes. But, they fail to extract the conditions like $\{experience > 5\}$ for discrete data attributes. The supervised learning approach we present can extract such conditions and works for both categorical and non-categorical attributes.

In the next section, we give some necessary background and definitions required to explain our approach.

3. Background and Preliminaries

In this section, we briefly give an overview of the concepts and definitions of the components of the RBAC and ABAC models.

RBAC works on the concept of role. The role is analogous to the user's job profile in an organization. For example *cashiers, accountants*, etc. A role is the set of

permissions needed to carry out the software activities to perform the job. Permission is the element comprising of (*o, op*). For example; a user with permission (*evaluate, grade-sheet*) can evaluate the grade sheet of the student. The set of such permissions is assigned to a role and the set of roles is assigned to users. RBAC system maintains the information about permission assignment and role assignment in the *permission-to-role assignment* relation and the *user-to-role assignment* relation respectively. ABAC works on the concept of user, object, and environment attributes as described in the previous section. Table 1 gives the difference between RBAC and ABAC concerning the access control mechanism.

Table 1. RBAC Vs ABAC

Sr. No	RBAC	ABAC
1	Access to system resources is controlled by roles.	Access to system resources is controlled by attribute-based policies
2	Access is granted in terms of the group of permissions. (i.e. role)	Access is granted single permission-wise.
3	To assign permission to the user, a role containing that permission is assigned to the user manually.	Permission is granted to the user at the time of request based on the values of the user, object, and environmental attributes.
4	Permissions are granted in advance to the request to access the permission is made.	Permission is granted at run-time.
5	Access control information is maintained in terms of user-to-role assignment and role-to-permission assignment relations	Access control information is maintained in terms of attribute-based access control policies per permission.

Table 2 gives the symbols and their meanings used to describe the concepts of RBAC and ABAC models and the model we are presenting.

Table 2. Symbols to Describe the Concepts of RBAC and ABAC

Name	Symbol	Description
Object	O	Set of system objects like file, database record, loan account information, etc.
Operation	OP	Set of operations like read, write, approve the loan, etc.
Permission	p	(o, op) an element representing an operation, op is allowed on the object, o.
Permission Set	P	The set of all possible permissions in the system. i.e. $P = \{p_1, p_2, \ldots, p_n\}$
Role	r	Set of permissions required to perform a particular task. i.e. $r = \{p_1, p_2, \ldots, p_k\}$ where $k<=n$
Role Set	R	Set of all possible roles in the system. i.e. $R=\{r_1, r_2, \ldots, r_m\}$
Attribute	a	User, object, or environment attribute
Conditional Value	v	The conditional value of the attribute that needs to satisfy. It may be numeric, character, or string
Relation Condition	c	$a \odot v$
Conditional Operator	$\odot$	$< \mid \leqslant \mid = \mid > \mid \geqslant$
ABAC Rule	ρ	$c_1 \wedge c_2, \ldots, \wedge c_p$
Rule Set	RS	Set of ABAC Rules

Definition 1: (**User-to-Role Assignment Relation**). A *user-to-role assignment relation* is defined as a set of tuples $UA = \{< u, r > \mid u \in U \text{ and } r \in R\}$. A tuple $< u, r >$ is that the role r is assigned to the user u.

Definition 2: (**Permission-to-Role Assignment Relation**). A *permission-to-role assignment relation* is defined as a set of tuples $PA = \{< p, r > \mid p \in P \text{ and } r \in R\}$. A tuple $< p, r >$ is that the permission p is assigned to the role r.

Definition 3: (**RBAC Transaction Log**). A *transaction log* of the RBAC system is a set of tuples $L = < u, r, p, t, d >$. Where t is a timestamp or any contextual information about the transaction like the location of the user performing transaction and d is the decision for the transaction (*permit* /*deny*).

Example 1: < *John*, *Evaluator*, (*evaluate, grade-sheet*), *t1*, *permitted* > indicates the user *John* with the role *Evaluator* had acquired the permission (*evaluate, grade-sheet*) at timestamp *t1*

Definition 4: (**ABAC Rule**). An ABAC *rule* is a set of conditions $\varphi = \{c_i | c_i = a_i \odot v_i\}$, where $a_i \in \{A_u \cup A_o \cup A_e\}$ and c_i is the relation condition for corresponding user, object and environmental attributes. It is interpreted as a conjunction $c_1 \wedge c_2 \wedge, \ldots, \wedge c_n$. A rule gets satisfied if it evaluates to true.

f_u and f_o are the user and object attribute value functions that return values for the user attributes, A_u, and object attributes, A_o respectively. V_{ua} and V_{oa} are the set of attribute values such that $V_{ua} = \{ v \mid \forall a_i \in A_u , v = f_u(a_i, u) \}$ *and* $V_{oa} = \{ v \mid \forall a_i \in A_o , v = f_o(a_i, o) \}$

Example 2: Suppose A_u = {designation, appointment, experience}, f_u(designation, John) = Assistant Professor, f_u(appointment, John) = Regular and f_u(experience, John) = 5 Then V_{Johna} = { Assistant Professor, Regular, 5}.

Definition 5: (**User Attribute Conditions**). User attribute conditions is a set of conditions, $C_{ua} = \{c_k \mid c_k = a_k \odot v_k, \forall a_k \in A_u\}$

Example 3: C_{ua} = {{designation ∈ [*Assistant Professor*, *Professor*]}, {*appointment* ∈ [*Adhoc*, *Regular*]}, {*experience* > 3}} indicates that the value of the user attribute *designation* can be either *Assistant Professor* or *Professor*, an *appointment* can be either *Contractual* or *Regular* and *experience* can be greater than *3* years.

Definition 6: (**Object Attribute Conditions**). Object attribute conditions is a set of conditions, $C_{oa} = \{c_m \mid c_m = a_m \odot v_m, \forall a_m \in A_o\}$

Definition 7: (**Environment Attribute Conditions**). Environment attribute conditions is a set of conditions, $C_{ea} = \{c_q \mid c_q = a_q \odot v_q, \forall a_q \in Ae\}$

With the above definitions and RBAC and ABAC concepts discussed, the next section describes our supervised machine learning approach for constructing ABAC rules.

4. The Proposed Supervised Machine Learning Approach

We present the mechanism to extract the attribute-based conditions from the existing RBAC system that are used to construct ABAC rules. It reduces the efforts of a system administrator while transforming the RBAC system into ABAC. Our approach uses the permission-to-role assignment relation, user-to-role assignment relation, and transaction log for this purpose. Tables 3 and 4 show the sample representation of user-to-role assignment relation and role-to-permission assignment relation respectively.

Table 3. Sample User-to-Role Assignment Relation

User	Roles
U1	R1, R3, R7
U2	R1, R2, R5, R8
U3	R3, R4, R6
U4	R3, R5, R7
U5	R1, R3, R8
U6	R2, R4, R6
U7	R2, R5,R7,R8
U8	R5,R7,R8
U9	R1, R2, R5, R7, R8
U10	R3, R4, R6, R7

Table 4. Sample Permission-to-Role Assignment Relation

Role	Permissions
R1	P1, P2, P3, P5
R2	P1, P2, P6, P7, P8
R3	P2, P5, P7, P8
R4	P2, P3, P4, P8
R5	P1, P3, P6, P7
R6	P3, P4, P7, P8
R7	P1, P4, P5, P6, P7, P8
R8	P3, P4, P5, P6

The roles of the RBAC system classify the users into various security classes. In ABAC, attribute-based rules classify the users into one of the security classes. Attribute conditions defined in the ABAC rule help to classify the users and decide whether the given permission can be granted to them or not. We have used the J48 decision tree machine algorithm to extract user, object, and environment attribute relation conditions required to construct the ABAC rule. Three separate decision trees are constructed to extract user, object, and environment attribute relation conditions respectively.

4.1. ABAC Rule Extraction Problem

The ABAC Rule extraction problem can be stated as:

Given user-to-role assignment relation (*UA*), permission-to-role assignment relation (*PA*), and transaction log (*L*); extract attribute condition set C_{ua}, C_{oa}, and C_{ea} and construct ABAC rules using them.

Data sets required for constructing decision trees are generated by using relations *UA, PA,* and system log entries. The next section depicts how these data sets are generated.

4.2. Generating Data Set for Machine Learning Process

Given the relation UA, a set of users to whom the role r is assigned is constructed. This set is constructed as follows:

$$U_r = \{u \mid \forall_u \in U, \forall_{(u_1, r)} \in UA, u = u_1\} \quad (1)$$

Table 5 shows the sample of U_r concerning the sample UA as shown in Table 3.

Given U_r, user attribute value data of all users, $u \in U_r$, is extracted by using function f_u, and the data set with tuples as below is generated.

$$T_u = \{ (V_{ua}, r) \mid (u, r) \in U_r\} \quad (2)$$

Table 5. Sample Ur Relation

Role	Users
R1	U1, U2, U5, U9
R2	U2, U6, U7, U9
R3	U1, U3, U4, U5, U10
R4	U3, U6, U10
R5	U2, U4, U7, U9
R6	U3, U6, U10
R7	U1, U4, U7, U8, U9, U10
R8	U2, U5, U7, U8, U9

Example 4: Suppose (*John, Evaluator*) $\in U_r$, then the corresponding tuple in T_u will be < *Assistant Professor*, *Regular*, *5*, *Evaluator* >

Assume $A_u = \{a_1, a_2, a_3, a_4\}$ are the overall user attributes in the system. Table 6 represents the set of tuples in T_u for the users to whom role *R1* is assigned (Table 5).

Table 6. Sample Tu Dataset

User	a_1	a_2	a_3	a_4	Role
U1	v a_{11}	v a_{12}	v a_{13}	v a_{14}	R1
U2	v a_{21}	v a_{22}	v a_{23}	v a_{24}	R1
U5	v a_{51}	v a_{52}	v a_{53}	v a_{54}	R1
U9	v a_{91}	v a_{92}	v a_{93}	v a_{94}	R1

Similarly, given the relation PA, object attribute value data for the objects o in $p \in P$ is extracted using f_o, and the data set with the tuples as below is generated.

$$T_o = \{(V_{oa}, p) \mid o \text{ is in } p \in P\} \quad (3)$$

Given the transaction log L, environment attribute value data is extracted from log entry, and data set with the tuples as below is generated.

$$T_e = \{(t, p) \mid (t, p) \text{ is in } l \in L\} \quad (4)$$

Once the datasets are ready, decision trees are constructed to extract the attribute relation conditions for generating ABAC rules. The decision tree construction process is explained in the next section.

4.3. Extraction of Relation Conditions

The decision tree classifier is applied to the above-generated data sets and the three separate decision trees for the user, object, and environmental attributes are constructed. The role is considered a categorical attribute for building the user classifier while permission is considered a categorical attribute for building the object and environment classifiers. Relation conditions are then extracted from decision rules provided by decision trees. The steps of the process for extracting relation conditions are summarized in algorithm1.

In Lines 1, 2, and 3 of the algorithm, decision trees are constructed by applying a decision tree classifier on the training datasets T_u, T_o *and* T_e respectively. Loop in lines 4 to 6 extracts user attribute conditions from decision tree TR_u while loop in lines 7 to 10 extracts object and environment attribute conditions from decision trees TR_o and TR_e respectively.

Extracted attribute conditions can now be useful to construct ABAC rules. The next section explains the rule generation process.

4.4. Generating ABAC Rules

Given user, object, and environment attribute relation conditions, the ABAC rules are constructed as per the steps summarized in algorithm 2.

Algorithm 1: Extracting Attribute Relation Conditions

Input: T_u, T_o, T_e, P, R
Output: $C_{ua}[]$, $C_{oa}[]$, $C_{ea}[]$

1. $TR_u \leftarrow$ DecisionTreeClassifier(T_u)
2. $TR_o \leftarrow$ DecisionTreeClassifier(T_o)
3. $TR_e \leftarrow$ DecisionTreeClassifier(T_e)
4. *for all* $r_i \in R$
5. $C_{ua}[r_i] \leftarrow$ ExtractUserAttCond(TR_u)
6. *end for*
7. *for all* $p_i \in P$
8. $C_{oa}[p_i] \leftarrow$ ExtractObjAttCond(TR_o)
9. $C_{ea}[p_i] \leftarrow$ ExtractEnvAttCond(TR_e)
10. *end for*
11. *return* $C_{ua}[]$, $C_{oa}[]$, $C_{ea}[]$
12. *end procedure*

Algorithm 2: Constructing ABAC Rules

Input: $C_{ua}[]$, $C_{oa}[]$, $C_{ea}[]$, R, P
Output: RS
1. $RS \leftarrow \Phi$
2. $\rho_k \leftarrow \Phi$
3. *for all* $r_i \in R$
4. *for all* $p_j \in r_i$
5. $\rho_k \leftarrow \rho_k \cup C_{ua}[r_i] \cup C_{oa}[p_j] \cup C_{ea}[p_j]$
6. $RS \leftarrow RS \cup \rho_k$
7. *end for*
8. *end for*
9. *return RS*
10. *end procedure*

Algorithm 2 generates the rules at the primary level. Generated rules can then be pruned if necessary by the system administrator. Administrators can also eliminate unnecessary rules from the rule set.

5. Results and Discussion

To evaluate the effectiveness of our algorithm on discrete data attributes and to show how relation conditions with relational operators are extracted, we have used the synthesized data sets for *UA, PA, and transaction log*. Implementation of these data structures and the data fields in the transaction log record vary from application to application. We have used general formats to demonstrate the working of the approach. Table 7 represents the sample user attribute value data we have used for experimentation. Entries in the table are the sample tuples generated by equation (2) as stated in the previous section.

Table 7. Sample Training Dataset of User Attribute Value Data

UID	DES	APPT	EXP	ROLE
1	AP	R	5	EV
2	AP	R	4	EV
3	AP	A	2	ST
4	P	R	10	PS
5	AP	A	5	ST
6	AP	R	8	PE
7	AP	R	6	PE
8	P	R	11	PS
9	AP	R	5	EV
10	AP	R	3	ST
11	AP	R	2	ST
12	AP	R	4	EV

Where *UID = User ID*, *APPT = Appointment*, *DES = Designation*, *EXP = Experience*, *A = Adhoc (Contractual)*, *R = Regular*, *AP = Assistant Professor, P = Professor*, EV = Evaluator, *ST = Subject Teacher*, *PE = Practical Examiner*, *PS = Paper Setter*

Rows in the table represent user information and the role assigned to him. A set of users with the same role forms a one-user class. For example; all the users to whom the role evaluator has been assigned belong to the evaluator class. The user information belonging to one user class forms a particular range(or set) of values for each data attribute. For example; user data values of the users belonging to the evaluator class will form one range(or set) of values for the data field experience while forming another range for other classes. It is difficult to extract such ranges from a large set of records manually. A decision tree helps to extract these data value ranges or conditions automatically.

The decision tree shown in Fig. 1 is constructed after the application of the decision tree classifier on the above data set with "*Role*" as a class index attribute.

```
EXP <= 6
| EXP <= 3: ST
| EXP > 3
| | APPT = R: EV
| | APPT = A: ST
EXP > 6
| EXP <= 9
| | DES = AP: PE
| | DES = P: PS
| EXP > 9: PS
```

Figure 1. Decision Tree Output

The decision tree rules shown in the decision tree of the figure provide the attribute relation conditions exactly similar to the conditions required to construct ABAC rules. The output of the decision tree classifier shows how conditions with inequalities for continuous attribute "*EXP*" have been extracted. One of the decision tree rules for the user class *PS* is as follows.

$\{DES = P\}$ *and* $\{APP = R\}$ *and* $\{EXP > 6\} \longrightarrow \{userClass = PS\}$

From the above decision tree rule, the following user attribute relation conditions are extracted to construct the ABAC rule for all the permissions belonging to role *PS*.

$$C_{ua}[PS] = \{\{Des = P\}, \{App = R\}, \{Exp > 6\}\}$$

Thus our supervised learning-based approach can extract relation conditions for discrete data attributes. The use of *user-to-role* relation for extraction of the rules provides an accurate user-role relationship compared to

approaches that use system logs for rule extraction. Due to supervised learning, the number of rules extracted would be more accurate compared to the unsupervised learning-based approach. Table 8 gives the comparison of our approach with the approaches discussed in the related work section.

As the supervised learning algorithm works on data set containing class labels, the user classification done by it is more accurate compared to unsupervised learning. Accurate classification leads to generating accurate attribute conditions. The information contained in the system log may be incomplete leading to generating an inaccurate number of rules. Our approach is supervised learning based and uses user-to-role and permission-to-role relations along with system log data to generate a dataset for the learning process. Similarly, the use of a decision tree classifier makes our approach applicable for discrete data attributes.

Table 8. Our ApproachesVs Other's Approaches

Approach	Technique	Training Data	Handles Discrete Data
Amani et al.[1]	Association Mining	System Log	×
Zhongyuan and Scott [27]	Association Mining	Access Control List	×
Carlos et al. [2]	Association Mining	Sparse Log	×
Matthew W Sanders and Chuan Yue [17]	Association Mining	System Log	×
Leila et al. [14]	Unsupervised Learning	System Log	×
Ours	Supervised Learning	System Log, *UA*, and *PA* Relations	✓

6. Conclusion

We have presented a supervised learning-based approach to extract attribute relation conditions from the existing RBAC system to construct ABAC rules at the primary level to transform them into an ABAC system. We have used user-to-role relation, permission-to-role relation, and system log to extract the conditions. The approach is effective and can extract attribute relation conditions for both categorical and discrete data attributes. However, our approach does not consider the multi-value attributes of the system, and it is mainly useful for transforming existing RBAC systems into ABAC systems.

References

[1] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. Communications of the ACM. 1976; 9(8):461–471.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role based access control models. Compute. 1996; 29(2):38–47.

[3] R. S. Sandhu and P. Samarati. Access control: principle and practice. IEEE communications magazine. 1994; 32(9) :40– 48.

[4] R. S. Sandhu. Lattice-based access control models. Computer. 1993; 26(11): 9–19.

[5] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations. Technical report, MITRE CORP BEDFORD MA(MAC). 1993.

[6] M. Beckerle and L. A. Martucci. Formal definitions for usable access control rule sets from goals to metrics. Proceedings of the Ninth Symposium on Usable Privacy and Security; 24 July; New York, NY, United States: ACM; 2013. p. 1-11.

[7] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security. 2001; 4(3): 224-274.

[8] David F. Ferraiolo, D. Richard Kuhn and Ramaswamy Chandramouli. Role Based Access Control. Second Edition. Artech House Inc, Norwood. 2007.

[9] Erkan et al. Application of Attribute Based Access Control Model for Industrial Control Systems. International Journal of Computer Network and Information Security. 2017; 9(2):12-21

[10] Depavath Harinath and P. Satyanarayana. A Review on Security Issues and Attacks in Distributed Systems. Journal of Advances in Information Technology. 2017; 8(1):1-8.

[11] Hyun-Jin Kim and Im-Yeong Lee. A study on a secure single sign-on for user authentication information privacy in Distributed computing environment. Journal of Communication Networks and Distributed Systems. 2017; 19(1):28-45.

[12] S. Hachana, N. Cuppens-Boulahia, and F. Cuppens. Role mining to assist authorization governance: How far have we gone? International Journal of Secure Software Engineering (IJSSE). 2112; 3(4):45–64.

[13] Coyne Ed. and Timothy R. Weil. ABAC and RBAC: Scalable, flexible, and auditable access management. IT Professional, IEEE Computer Society. 2013; 15(3):14-16.

[14] Dipmala Salunke, Anilkumar Upadhyay, Amol Sarwade, Vaibhav Marde and Sachin Kandekar. A survey paper on Role Based Access Control. International Journal of Advanced Research in Computer and Communication Engineering. 2013; 2(3):1340-1342.

[15] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, et al. Guide to Attribute Based Access Control (ABAAAC) definition and considerations (draft). NIST special publication. 2014; 800(162).

[16] X. Jin, R. Krishnan, and R. S. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. 26th Conference on Data and Applications Security and Privacy(DBSec); July; Paris, France; 2012; p. 41–55.

[17] H. Takabi and J. B. Joshi. Stateminer: an efficient similarity-based approach for optimal mining of role hierarchy. Proceedings of the 15th ACM symposium on Access control models and technologies; June 9-11; Pittsburgh, Pennsylvania, USA: ACM; 2010; p. 55–64.
[18] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with semantic meanings. Proceedings of the 13th ACM symposium on Access control models and technologies; June 11-13; Estes Park CO USA:ACM; 2008; p. 21–30.
[19] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, and J. Lobo. Mining roles with multiple objectives. ACM Transactions on Information and System Security (TISSEC). 2010; 13(4):1-35.
[20] J. Vaidya, V. Atluri, and J. Warner. Role miner: mining roles using subset enumeration. Proceedings of the 13th ACM conference on Computer and communications security; Oct 30-Nov 03; Alexandria Virginia USA:ACM; 2006; p. 144–153.
[21] Q. Ni, J. Lobo, S. Calo, P. Rohatgi, and E. Bertino. Automating role-based provisioning by learning from examples. Proceedings of the 14th ACM symposium on Access control models and technologies; June 3-5; Stresa Italy:ACM; 2009; p. 75–84..
[22] Z. Xu and S. D. Stoller. Algorithms for mining meaningful roles. Proceedings of the 17th ACM symposium on Access Control Models and Technologies; June 20-22; Newark New Jersey USA:ACM; 2012; p. 57–66.
[23] L. Karimi, M. Aldairi, J. Joshi and M. Abdelhakim. An Automatic Attribute Based Access Control Policy Extraction from Access Logs. IEEE Transactions on Dependable and Secure Computing. 2022; 19: 2304-2317.
[24] Matthew W Sanders and Chuan. Mining Least Privilege Attribute Based Access Control Policies. ACSAC '19: Proceedings of the 35th Annual Computer Security Applications Conference; December 9-13; San Juan Puerto Rico USA:ACM; 2019; p. 404–416.
[25] Z. Xu and S. D. Stoller. Mining attribute-based access control policies from rbac policies. Emerging Technologies for a Smarter World (CEWIT), 2013 10th International Conference and Expo; Oct 21-22; Melville, NY:IEEE; 2013; p. 1–6.
[26] Z. Xu and S. D. Stoller. Mining attribute-based access control policies from logs. IFIP Annual Conference on Data and Applications Security and Privacy; July 14-16; Vienna, Austria:Springer; 2014;. p. 276–291.
[27] Z. Xu and S. D. Stoller. Mining attribute-based access control policies. IEEE Transactions on Dependable and Secure Computing. 2015; 12(5):533–545.
[28] Amani Abou Rida, Nour Assy, Walid Gaaloul. Extracting Attribute-Based Access Control Rules From Business Process Event Logs. Proceedings of the 2nd International Conference on Big Data and Cyber-Security Intelligence; December 16-17; Versailles, France; 2019; p. 38-45.
[29] Carlos Cotrini, Thilo Weghorn, David Basin. Mining ABAC Rules from Sparse Logs. IEEE European Symposium on Security and Privacy (EuroS&P); April 24-26; London, UK:IEEE; 2018; p. 31-46.