Chapter

Role of Access Control in Information Security: A Security Analysis Approach

Mahendra Pratap Singh

Abstract

Information plays a vital role in decision-making and driving the world further in the ever-growing digital world. Authorization, which comes immediately after authentication, is essential in restricting access to information in the digital world. Various access control models have been proposed to ensure authorization by specifying access control policies. Security analysis of access control policies is a highly challenging task. Additionally, the security analysis of decentralized access control policies is complex because decentralization simplifies policy administration but raises security concerns. Therefore, an efficient security analysis approach is required to ensure the correctness of access control policies. This chapter presents a propositional rule-based machine learning approach for analyzing the Role-Based Access Control (RBAC) policies. Specifically, the proposed method maps RBAC policies into propositional rules to analyze security policies. Extensive experiments on various datasets containing RBAC policies demonstrate that the machine learning-based approach can offer valuable insight into analyzing RBAC policies.

Keywords: role-based access control, security analysis, propositional rule, safety analysis, reachability analysis

1. Introduction

Access control [1] ensures secure access to resources, devices, and data through policies. It regulates who can access which computing environment and its components. There are various access control models, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), etc., that can specify and enforce access control policies. Among these, RBAC [2] is a widely adopted access control model that groups job functions into roles to simplify the administration. In RBAC, permissions are actions on objects assigned to roles instead of users. Therefore, a user can get specific permission only if the user is a member of the role to which the permission is assigned. RBAC can be described as a 6-tuple access control model, and its components are as follows.

- U, P, R, and Sessions represent a set of users, permissions, roles, and sessions, respectively. Sessions are not considered in the proposed approach because they do not influence security analysis.

- PA denotes permissions to role assignments and is represented as PA $\subseteq$ P $\times$ R.
- UA denotes users to role assignments and is represented as UA $\subseteq$ U $\times$ R.
- Each session is mapped to a single user and is represented as S$\rightarrow$ U.
- Each session can have one or more roles and is represented as Session $\rightarrow 2^R$.
- The role hierarchy is defined as a partial order relation on roles and is represented as RH $\subseteq$ R $\times$ R.

The RBAC components (UA, PA, RH) determine whether users can access a particular resource, system, or data. Therefore, any change in these components would take a system to a new state. Hence, verifying whether it is safe is necessary before moving the system into a new state.

Security analysis aims to answer critical questions, such as whether a state is reachable to at least one user, whether all the reachable states satisfy security property, etc. An undesired state would be one in which an authorized user does not get access despite being entitled to it or an unauthorized user gains access.

One should consider various security properties before deploying a system. In this paper, we focus on safety and reachability properties that are described as follows:

Safety Property:

- Whether a user u can access permission p.
- Whether a user u can perform an access right r on an object o.

Reachability Property:

- Whether a role R is reachable to a user u.
- Whether a permission p is reachable to a role R.

If the evaluation outcome of a safety query mentioned above is no, then the system is safe. In contrast, if the evaluation outcome of a reachability query is yes, then the system is reachable.

The rest of the chapter is organized as follows. Previous works related to this research are presented in Section 2. Section 3 explains the proposed model, whereas Section 4 describes the security analysis of RBAC policies. Section 5 presents the result analysis and parameters used in the model, while Section 6 concludes the work and provides future research directions.

2. Related work

This section reviews the literature on recent applications, administration, and security analysis of RBAC policies.

Over the last few years, various access control models have been proposed. Among these, RBAC [2] is one of the well-adapted access control models. In RBAC, permissions are available to users according to their membership in specific roles. A large group of users can be grouped into roles to access resources according to permission assigned to the roles. Therefore, users should have a specific role to gain the required permissions for a task. The role-role relation enables delegation of authority and separation of authority. The main advantage of RBAC is the ease of policy administration.

2.1 Applications of RBAC

Recently, Kim et al. [3] have demonstrated RBAC usage in video surveillance using smart contracts, whereas Shaobo et al. [4] used RBAC to ensure fine-grained access to electronic health records. Additionally, Gurucharansing et al. [5] demonstrated the use of RBAC in specifying large-scale application access control policies. A Blockchain-based RBAC model with separation of duty presented by Ok-Chol et al. [6].

2.2 Administration of RBAC

Despite the advantages of RBAC, the administration of RBAC is complex and crucial for its proper management. Sandhu et al. [7] have presented the formal definition, intuition, and motivation of a role-based model for the administration of RBAC, ARBAC97 (administrative RBAC). The primary basis for the model is the simplification of administration along with scalability and administrative convenience. The components of the ARBAC97 model are user-role assignment (URA97), permission-role assignment (PRA97), and role-role assignment (RRA97).

2.3 Security analysis of RBAC

Apart from the administration, security analysis of the RBAC policies needs to be considered seriously. Alpern et al. [8] have formally defined safety and liveness security properties. Additionally, a topological characterization of both properties is also given. Their work captures all the main distinctions of the security properties. Koch et al. [9] have proposed safety state change rules where the RBAC states are posed as graph formalism in the RBAC model. Safety is defined as if a provided graph can become a subgraph of another graph. They have demonstrated that safety is decidable because a state change rule cannot simultaneously add and remove components to a graph. The proposed notion of safety captures the general notion but needs to cover bounded safety. Phillips et al. [10] proposed an access control model for servers, databases, inter-operating legacy, etc. Their work presents several theorems and lemmas to validate integrity and security. The combination of security and integrity ensured the proposed approach's liveness and safety. This approach does not consider the constraints of the RBAC model.

Li et al. [11] have proposed a security analysis approach using role-based trust management language for RBAC. They defined the problems related to security analysis and presented a way to represent and capture several security properties in a complex RBAC system. Specifically, two problem classes, namely AAR (Assignment and Revocation) and AATU (Assignment and Trusted Users), are discussed in the paper. The approach is based on reducing the two problem classes into another similar role-based trust-management language. This way, a relationship between the RT framework and RBAC is established. The approach produces efficient algorithms to solve significant queries. They demonstrated that several problems in security analysis need to be more concrete and intractable.

Jha et al. [12] performed the security analysis of the URA97 component of the ARBAC97 model using model-checking and logic programming approaches. Their work results demonstrate that the logic programming approach is better for many roles than the model-checking approach. Rakkay et al. [13] performed the security analysis by modeling and analyzing RBAC policies with the help of CPN tools and Colored Petri Nets (CP-Nets). The approach elaborates on the CP-Net model, which explains a generic access control structure based on RBAC policies. The significant advantage of CPN tools and CP-Nets is to provide an analytical framework and a graphical representation, which the security administrators use to understand why some permissions are denied or granted. Also, the framework and model are used to verify the security constraints.

Ferrera et al. [14] have proposed an approach to verify RBAC security policies using an abstraction-based tool. The proposed method converts data into imperative programs and performs security analysis. VAC tool was used to convert policies into crucial programs. An interval-based static analysis was carried out on the critical programs to show the correctness of policies. Martin et al. [15] have proposed a data-mining methodology to infer access control policies. The proposed approach is based on a tool developed for automatically generating requests, evaluating the requests to obtain the responses, and finally, using machine learning on the response-request pair to infer policy properties. The tool assists a user in identifying those requests, which can identify mistakes in the policy specification.

Most approaches mentioned above need to consider the overhead of translation of access control policies from one format (say XACML) to a specific format to perform security analysis. To address this, Singh et al. [16] have presented a framework that enables the specification and enforcement of heterogeneous access control policies, such as RBAC and ABAC, as data in the Database. Additionally, Singh et al. [17] have also presented a novel methodology for analyzing the security properties of heterogeneous access control policies. The proposed methodology models policies as facts using Datalog and analyses them through the μz tool in the presence of the administrative model. In addition, an approach to analyzing unified access control policies is also presented in [18] that captures policies as data in the Database.

It can be observed from the above literature survey that it is the first attempt to analyze access control policies using a machine learning-based model. The following section presents the proposed machine learning-based approach for analyzing RBAC policies.

3. Proposed security analysis approach

This section presents the approach for analyzing the security properties of RBAC policies. The proposed approach uses a rule-based machine learning algorithm to map RBAC policies into propositional rules. **Figure 1** shows the proposed model, and the description of its components is as follows.

3.1 Extraction of RBAC policy data from the unified database Schema

Generally, RBAC policies are specified using XACML, but we captured them as data in a unified database schema presented in [19]. There can be various combinations of RBAC policy data, but we have considered the following.

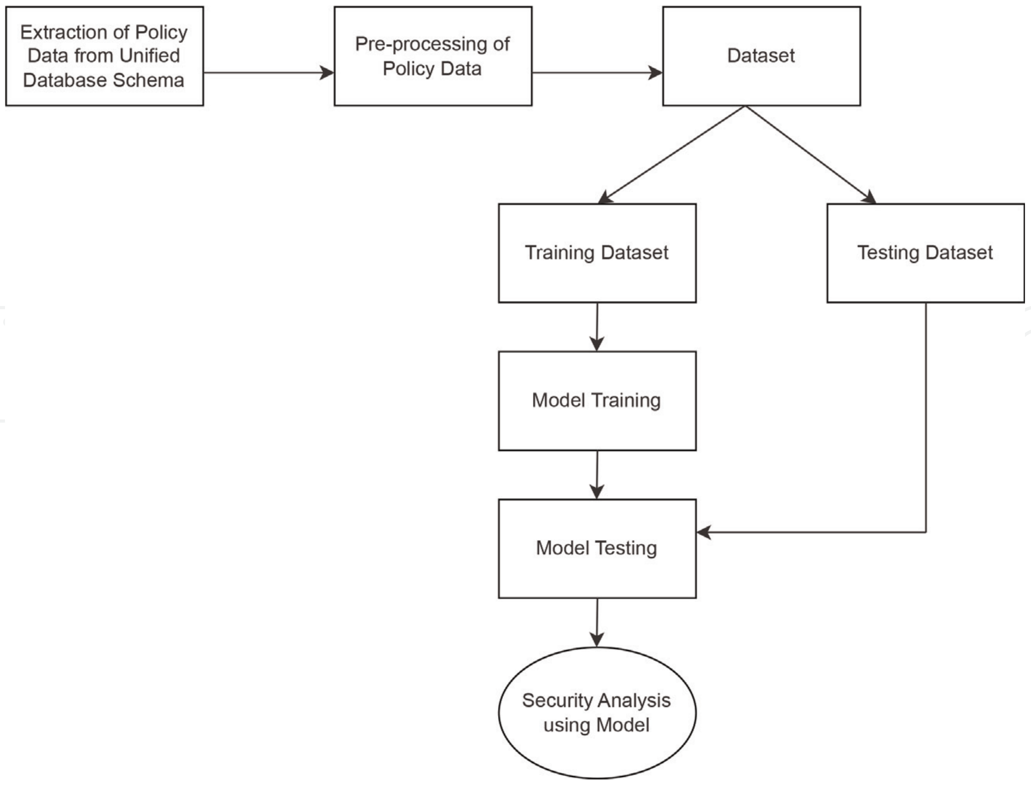
Extraction of Policy Data from Unified Database Schema
Pre-processing of Policy Data
Dataset
Training Dataset
Testing Dataset
Model Training
Model Testing
Security Analysis using Model

Figure 1.
Proposed model.

- Roles-Users: It captures the direct and indirect (through role hierarchy) association of roles with users.
- Permissions-Roles: It captures the direct and indirect (through role hierarchy) associations of permissions with roles.
- Users, Roles, and Permissions: It captures direct and indirect (through role hierarchy) associations of permissions with users through the role(s).
- Users, Roles, Rights, and Objects: It captures direct and indirect (through role hierarchy) associations of objects and rights with users through the role(s).

In the above combinations, attributes such as role, user, permission, object, and right act as features. Moreover, an extra feature, named label, has also been added. The above policy data combinations are fed to the following subsection for further processing.

3.2 Pre-processing of policy data to generate a dataset

In pre-processing, the policy data obtained in Subsection 3.1 was passed as input to AWK [20] for text manipulation and labeling. AWK is a Linux/Unix text manipulation utility that searches and substitutes text. Its output is a file containing only valid entries. Each valid entry represents authorized access and is marked as a safe state.

For security analysis, it is essential to consider valid and invalid entries. Therefore, another file containing both valid and invalid entries was created using the cartesian product of the attributes. Unlike valid entries, invalid entries represent unsafe states that indicate unauthorized access. Both valid and invalid entries were combined to form a broad system with safe and unsafe states. The valid and invalid entries were labeled 'Permit' and 'Deny', respectively.

The step-by-step process to create the dataset is as follows:

i. First, it takes a policy data file, say File 1, containing valid entries.

ii. Then generates another file, say File 2, through the Cartesian product of attributes.

iii. Next removes the common entries between File 1 and File 2 from File 2 and labels each entry as Deny.

iv. Labels all the entries of File 1 as Permit.

v. Combines the two files to produce a file containing both 'Permit' and 'Deny' labeled entries.

vi. Then, converts the file obtained above to a CSV file.

vii. Finally, the CSV file is converted to an ARFF file.

The above process is used to create the testing and following training datasets.

- Test dataset with all 'Permit' entries.
- Test dataset with all 'Deny' entries.
- Test dataset with a combination of 'Permit' and 'Deny' entries.

The datasets created in this subsection are used in the following subsections for training and testing the proposed model.

3.3 Proposed model and its training

This subsection describes the proposed model and its training using the dataset created in Subsection 3.2. We use a rule-based machine learning algorithm to create the model that takes the labeled dataset as input and maps it into propositional rules.

In the rule-based algorithm, a machine learning process identifies, evolves, or learns 'rules' to apply, manipulate or store.

The structure of a propositional rule is as follows:

$$\text{If}\,(\text{attribute1} < \text{a})\ \text{and}\ (\text{attribute2} > \text{b}) \dots \dots \text{and}\ (\text{attributen} < \phi\,) = \, > \text{class label}$$

Where a, b, $\dots$, and ϕ are the values the algorithm identifies from the policy attribute1, attribute2, $\dots$, and attributen, respectively, to generate the rule. In the above rule, the left side denotes the pre-condition or antecedent, a combination of attributes, whereas the right side shows the rule's consequent/class label.

We develop the model using **JRipper** [21] algorithm that implements a propositional rule learner known as RIPPER. This rule learner was proposed by William W.

Cohen and is based on a very effective and common technique of REP found in the decision tree algorithms.

The rule learner divides the training dataset into pruning and growing. First, the growing set is used to form an initial rule set with the help of some heuristic approach. Later, the large rule set is simplified repetitively using different pruning operators. The pruning operator with the most error reduction is selected at every simplification stage. The terminating point of simplification is when none of the operators reduces errors. It provides propositional rules for the training dataset used for classification.

The description of steps involved in the JRipper algorithm are as follows:

- **Initialization Stage:** In this stage, the rule set (RS) is initialized as RS={} for every class from most frequent to least frequent one.

- **Growing and Pruning Phase:** This phase repeats several steps until the error rate is below 50% or there is no positive example. In the growing phase, terms are added to a greedy rule to make the rule perfect. The rule is pruned incrementally in the pruning phase. The formula used to measure the pruning value is $2p / (p + n) - 1$, where p is the positive example covered by the rule, and n is the negative example covered by the rule.

- **Optimization Phase:** After identifying the rule set, two variants are created for every rule using random data, and then pruned. For the first variant, an empty rule is used. The addition of antecedents makes the second variant. The most petite description length is calculated for the original rule and its variants. If any residual positives are observed, they are used to identify more rules.

- **Delete Phase:** If the description length for any rule exceeds the limit, the rule gets deleted. The remaining rules are appended to form the final rule set.

We evaluate the model's performance using testing datasets in the following subsection.

3.4 Model testing

After training the model, the next step is to test the model. Testing datasets created in Subsection 3.2 are used to predicate the model's accuracy. Like the training dataset, the testing dataset contained policy data in ARFF file format. Every policy data is mapped to a predicate rule, then compared with the model's predicate rules. The outcome of the evaluation and accuracy is reported in Section 5.

In the following section, we use an example to demonstrate the security analysis of RBAC policies through the proposed model.

4. Analysis of security property

This section shows how the proposed model can be used to verify the safety and reachability properties of RBAC policies.

To understand the security analysis effectively, we consider an RABC system that consists of 20 users, 3 roles, and 30 permissions and the following UA, PA, and RH assignments:

Role={1,2,3}
User={1,2,3....20}
Permission={1,2,3,4....30}
RH={$3 \preccurlyeq 1$}
UA={({1,2..7},1), ({8,9..13},2), ({14,15..20},3)}
PA={({1,2..10},1), ({11,12..20},2), ({21,22..30},3)}

We create a model using the proposed approach for the above specification that contains propositional rules. The propositional rules correspond to some of the policies specified above are as follows:

- **Rule 1**: (User ID<=7) and (Role ID=1)=>Permit
- **Rule 2**: (User ID<=7) and (Role ID=2)=>Deny
- **Rule 3**: (User ID<=7) and (Role ID=3)=>Deny
- **Rule 4**: (User ID>=8) and (User ID<=13) and (Role ID=1)=>Deny
- **Rule 5**: (User ID>=8) and (User ID<=13) and (Role ID=2)=>Permit
- **Rule 6**: (User ID>=8) and (User ID<=13) and (Role ID=3)=>Deny
- **Rule 7**: (User ID>=14) and (User ID<=20) and (Role ID=1)=>Deny
- **Rule 8**: (User ID>=14) and (User ID<=20) and (Role ID=2)=>Deny
- **Rule 9**: (User ID>=14) and (User ID<=20) and (Role ID=3)=>Permit
- **Rule 10**: (Permission ID<=10) and (Role ID=1)=>Permit
- **Rule 11**: (Permission ID<=10) and (Role ID=2)=>Deny
- **Rule 12**: (Permission ID<=10) and (Role ID=3)=>Deny
- **Rule 13**: (Permission ID>=11) and (Permission ID<=20) and (Role ID=1) =>Deny
- **Rule 14**: (Permission ID>=11) and (Permission ID<=20) and (Role ID=2) =>Permit
- **Rule 15**: (Permission ID>=11) and (Permission ID<=20) and (Role ID=3) =>Deny
- **Rule 16**: (Permission ID>=21) and (Permission ID<=30) and (Role ID=1)=>Permit
- **Rule 17**: (Permission ID>=21) and (Permission ID<=30) and (Role ID=2)=>Deny
- **Rule 18**: (Permission ID>=21) and (Permission ID<=30) and (Role ID=3) =>Permit

Similarly, the propositional rules are specified for the remaining policies.

To perform the security analysis, we consider safety and reachability security properties defined in Section 1, and their analysis is as follows.

- **Security Analysis of Safety Property:** Test cases in the following form are passed to the model to analyze the safety property.
    - (User ID=5) and (Role ID=1) and (Permission ID=17)
    - (User ID=23) and (Role ID=2) and (Permission ID=27)
    - (User ID=9) and (Role ID=3) and (Permission ID=15)

The model classified all the test cases mentioned above as 'Deny' for the following reasons.

- User with USER ID=5 is not authorized to access Permission with Permission ID=17 through role with Role ID=1.
- User with USER ID=23 is not authorized to access Permission with Permission ID=27 through role with Role ID=2.
- User with USER ID=9 is not authorized to access Permission with Permission ID=15 through role with Role ID=3.

It can be noticed from the above analysis that the Role ID in each test case cannot provide permission to the user according to the rules present in the model. Thus, safety property satisfies.

- **Security Analysis of Reachability Property:** The test cases in the following form are considered to analyze the reachability property.
    - (User ID=5) and (Role ID=1)
    - (User ID=14) and (Role ID=3)
    - (User ID=9) and (Role ID=2)

The model classified all the test cases mentioned above as 'Permit' for the following reasons.

- User with USER ID=5 has role with Role ID=1.
- User with USER ID=14 has role with Role ID=3. For
- User with USER ID=9 has a role with Role ID=2.

It can be seen from the above analysis the Role ID in each test case is available to the user according to the rules present in the model. Thus, the reachability property holds.

The experimental analysis of the proposed model is demonstrated in the following section.

5. Experimental results and analysis

Several experiments were performed on the system having 64 GB RAM and an Intel Core i7 processor to observe the impact of various components of RBAC policies.

To evaluate the performance of the proposed model, we created three synthetic RBAC policy datasets shown in **Table 1** using Oracle 12c that capture policies as data. To reflect the real-world scenario, it can be observed from **Table 1** that the number of users and the number of objects were increased 100 times, whereas the number of rights was increased only two times. The number of permissions and roles were increased 200 times and four times, respectively. Additionally, the number of permission-role assignments was increased to 100, while the number of permissions and user-role assignments were increased to 200 times.

The following parameters were used to measure the performance of the proposed model:

- True Positive(TP) rate is the ratio of instances correctly classified for a class to the total number of instances.

$$\text{TP rate} = \text{TP}/(\text{TP} + \text{FN})$$

- False Positive(FP) rate is the ratio of negative events wrongly categorized as positive to the total number of actual negative events.

$$\text{FP rate} = \text{FP}/(\text{FP} + \text{TN})$$

- Precision is the ratio of instances that belong to that class to the total number of instances classified as that class.

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

- Recall is the ratio of instances classified as a given class to the actual number of instances that belong to that class.

Relation	Dataset 1	Dataset 2	Dataset 3
Users	75	750	7500
Objects	100	1000	10,000
Rights	5	5	10
Permissions	100	1000	20,000
Roles	10	20	40
Permission Role Assignment	100	1000	20,000
User Role Assignment	150	1500	15,000
Permission Object Assignment	100	1000	20,000

Table 1.
Number of entries for relations in the datasets.

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

- Confusion Matrix: It shows how many instances of a particular class are correctly or incorrectly classified.

- F-measure depends upon precision and recall.

$$\text{F-measure} = (2^{*}\,\text{precision}^{*}\,\text{recall})/(\text{precision} + \text{recall})$$

- k-fold cross validation: The training dataset is divided into k-sets. Of the k sets, the k-1 is used for training, and the remaining one is used for testing. It is repeated k times, and a different set is used for testing each time. After k iterations, the average error across k-trials is measured.

The following four models were trained and tested to classify instances as Permit or Deny using Datasets 1, 2, and 3. The value of k was kept at 10 for Dataset 1, while 2 for Datasets 2 and 3. The description of the models is as follows:

- URA is composed of users and roles.

- PRA consists of permissions and roles.

- URP is made up of users, permissions, and roles.

- UROR comprises users, objects, rights, and roles.

The following parameter values were obtained for the above models.

- **Accuracy of Model:** It shows how many instances of labeled data were correctly identified by the model in percentage. In other words, it shows a model's reliability in reflecting the RBAC policies.

- **Time to Build the Model:** It shows the time the algorithm takes to construct a rule set for the classifier from the labeled data set.

- **Accuracy of Test Results:** It shows how many entries of the test dataset were correctly classified by the classifier.

Table 2 shows the model's accuracy, the time to build a model, and the accuracy of test results. It can be observed from the table that model and test result accuracy are

Dataset	Accuracy of Model (%)				Time to Build the Model (Sec)				Accuracy of Test Results (%)			
	URA	PRA	URP	UROR	URA	PRA	URP	UROR	URA	PRA	URP	UROR
1	96.67	87.80	99.99	99.07	0.03	0.18	10.67	197.79	96.13	100	100	100
2	99.22	99.33	99.99	99.98	4.96	9.71	132.72	814.48	100	100	100	100
3	99.85	99.94	99.99	99.99	1315.62	56.97	446.73	672.63	100	100	100	100

Table 2.
Performance of models.

Dataset	Access	Predicted: Deny				Predicted: Permit			
		URA	PRA	URP	UROR	URA	PRA	URP	UROR
1	Actual: Deny	302	691	70,650	370,650	14	19	0	0
	Actual: Permit	11	103	1	35	423	187	4349	4315
2	Actual: Deny	10,633	17,070	218,000	224,991	27	30	0	9
	Actual: Permit	101	104	35	78	4249	2796	217,465	224,922
3	Actual: Deny	224,976	719,965	8,399,996	6,999,998	24	35	4	2
	Actual: Permit	429	480	0	1	74,571	79,520	8,710,000	7,059,999

Table 3.
Confusion matrix of models.

near about 100% for all three datasets. Additionally, there is no significant increase in time to build models for Datasets 1, 2, and 3.

The confusion matrix of models is shown in **Table 3**. From the table, it can be seen that most of the models predicted instances accurately. Therefore, it can be concluded that the proposed model is scalable and can be a viable option.

6. Conclusion

The security analysis of the RBAC policies has been performed through the Machine Learning-based model that uses the JRipper algorithm. The proposed model could map most policies correctly to rule sets for each classifier. The results show that the proposed model is highly reliable and efficient for the security analysis of RBAC policies. Additionally, it can also be observed that the model's efficiency has improved significantly due to an increase in the RBAC policy datasets. Thus, the proposed approach can be considered a viable solution for performing the security analysis of large policy sets.

In the future, the proposed model can be extended to analyze the other security properties of RBAC with and without an administrative model. Moreover, it can also be used to analyze RBAC extensions (such as TRBAC, ESTRBAC, etc.) security properties in the presence and absence of an administrative model.

Abbreviations

DAC	Discretionary Access Control
MAC	Mandatory Access Control
RBAC	Role-Based Access Control
ARBAC97	Administrative RBAC
URA97	User-Role Assignment
PRA97	Permission-Role Assignment
RRA97	Role-Role Assignment
ABAC	Attribute-Based Access Control
TRBAC	Temporal Role-Based Access Control
ESTARBAC	Extended Spatio-Temporal Role-based Access Control

AAR	Assignment and Revocation
AATU	Assignment and Trusted Users
XACML	Extensible Access Control Markup Language

Author details

Mahendra Pratap Singh
Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, Mangaluru, India

*Address all correspondence to: mahoo15@gmail.com

References

[1] National Institute of Standards and Technology, and National Security Agency. A Survey of Access Control Models. 2009. Available from: https://csrc.nist.gov/csrc/media/events/privilege-management-workshop/documents/pvm-model-survey-aug26-2009.pdf

[2] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role based access control models. IEEE Computer. 1996;**29**(2):38-47. Available from: https://ieeexplore.ieee.org/document/485845

[3] Kim J, Park N. Role-based access control video surveillance mechanism modeling in smart contract environment. Transactions on Emerging Tel Tech. 2022;**33**:e4227. DOI: 10.1002/ett.4227

[4] Zhang S, Yang S, Zhu G, Luo E, Xiang JZD. A fine-grained access control scheme for electronic health records based on roles and attributes. International Conference on Ubiquitous Security. 2022;**1557**:25-37. DOI: 10.1007/978-981-19-0468-4_3

[5] Sahani GJ, Thaker CS, Shah SM. Scalable RBAC model for large-scale applications with automatic user-role assignment. International Journal Communication Networks and Distributed Systems. 2022;**28**(1):120294. DOI: 10.1504/IJCNDS.2022.120294

[6] Ri OC, Kim YJ, Jong YJ. Blockchain-based RBAC Model with Separation of Duties constraint in Cloud Environment. arXiv. 2022. Available from: https://arxiv.org/abs/2203.00351

[7] Sandhu R, Bhamidipati V. Qamar Munawer: The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and System Security. 1999;**1999**:105-135. Available from: https://dl.acm.org/doi/10.1145/300830.300839

[8] Alpern B, Schneider FB. Defining liveness. Information Processing Letters. 1985;**21**(4):181-185. Available from: https://www.sciencedirect.com/science/article/abs/pii/0020019085900560

[9] Koch M, Mancini LV, Parisi-Presicce F. Decidability of safety in graph-based models for access control. In: Proceedings of the Seventh European Symposium on Research in Computer Security. 2002. pp. 229-243. Available from: https://link.springer.com/chapter/10.1007/3-h540-45853-0_14

[10] Phillips C, Demurjian S, Ting TC. Safety and liveness for an RBAC/MAC security model. In: Proceedings of the Data and Applications Security XVII. 2004. pp. 316-329. Available from: https://link.springer.com/chapter/10.1007/1-4020-8070-0_23

[11] Li N, Tripunitara MV. Security analysis in role-based access control. ACM Transactions on Information and System Security. 2006;**9**(4):391-420. Available from: https://dl.acm.org/doi/10.1145/1187441.1187442

[12] Jha S, Li N, Tripunitara M, Wang Q, Winsborough W. Towards formal verification of role-based access control policies. IEEE Transactions on Dependable and Secure Computing. 2008;**2008**:242-255. Available from: https://ieeexplore.ieee.org/document/4358710

[13] Rakkay H, Boucheneb H. Security analysis of role based access control models using Colored petri nets and CPNtools. Transactions on Computational Science IV. 2009;**2009**:

147-176. Available from: https://link.springer.com/chapter/10.1007/978-3-642-01004-0_9

[14] Ferrara AL, Madhusudan P, Parlato G. Security analysis of role-based access control through program verification. In: In the Proceedings of the IEEE 25th Computer Security Foundations Symposium. 2012. pp. 113-125. Available from: https://ieeexplore.ieee.org/document/6266155

[15] Martin E, Xie T. Inferring access-control policy properties via machine learning. In: In the Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks. 2006. pp. 1-4. Available from: https://ieeexplore.ieee.org/document/1631178

[16] Singh MP, Sural S, Vaidya J, Atluri V. Managing attribute-based access control policies in a unified framework using data warehousing and In-memory database. Computer & Security. 2019;**86**:183-205. Available from: https://www.sciencedirect.com/science/article/pii/S0167404819301166

[17] Singh MP, Sural S, Atluri V, Vaidya J. A role-based administrative model for administration of heterogeneous access control policies and its security analysis. Information Systems Frontiers. 2021;**2021**. Available from: https://link.springer.com/article/10.1007/s10796-021-10167-z

[18] Singh MP, Sural S, Atluri V, Vaidya J. Security analysis of unified access control policies. In: Proceedings of the International Conference on Secure Knowledge Management in Artificial Intelligence Era. 2019. pp. 126-146. Available from: https://link.springer.com/chapter/10.1007/978-981-15-3817-9_8

[19] Singh MP, Sural S, Atluri V, Vaidya J. Managing multi-dimensional multi-granular security policies using data warehousing. In: In the Proceedings of the International Conference on Network and System Security. 2015. pp. 221-235. Available from: https://link.springer.com/chapter/10.1007/978-3-319-25645-0_15

[20] Awk. Available from: http://www.grymoire.com/Unix/Awk.html

[21] Shahzad W, Asad S, Khan MA. Feature subset selection using association rule mining and JRip classifier. International Journal of Physical Sciences. 2013;**8**(18):885-896. Available from: https://academicjournals.org/journal/IJPS/article-abstract/22AC4CB27262