International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Recommender Algorithms for Adaptive Access Control Mechanism of Enterprise Cloud

Amardeep Kaur

Submitted: 02/02/2023 **Revised**: 19/03/2023 **Accepted**: 27/03/2023

Abstract: The purpose of this paper is to propose recommendations for defining access control policies for an enterprise cloud. These recommendations aim to introduce adaptability to changing requirements of enterprise users in access control mechanisms, with the goal of making a limited set of resources available to users of all roles who actually need them. The methodology involves developing recommender algorithms for redefining access control policies and improving resource utilization with the same set of resources. Implementation of the algorithms demonstrated a significant improvement in the availability of resources to desired users. The study's novelty lies in two aspects. First, it introduces dynamism in access control policies that are otherwise static in nature. Second, the access control policies help improve resource utilization with the same set of resources.

Keywords: *Adaptive Access Control; Enterprise Cloud; Recommender Algorithm; Resource Utilisation; Role Based Access Control*

1. Introduction

Enterprise Cloud Computing offers computing resources to an enterprise. It provides a secure computing environment through an authentication and access control mechanism. The primary purpose of access control is to have controlled access to the resources of the enterprise cloud, and they are static in nature. Such controlled access is for security reasons and also to restrict access to the resources if they are limited in number. The role-based access control (RBAC) [1] is the base algorithm of the present study. The RBAC relies on the access control policies defined for each role. The role is generally a function of the user designation and/or other assigned responsibilities of the user. In traditional RBAC, which is one of the popular and existing access control models, these policies are static, and they may result in the over availability and under availability of resources to roles because of a lack of assessment or strategy, and involve subjectivity in the formulation of access control policies for various roles. These may lead to underutilization of resources. Therefore, to introduce objectivity in the formulation of the definition of policies, with an intent to increase the availability of resources to the users of an enterprise, the present study suggests four recommendation algorithms to propose recommendations for access control policies to make them adaptive to the dynamic environment of the enterprise. The algorithms can introduce dynamism in the access control mechanism, and such dynamism ultimately increases the resource utilization of the enterprise. Such recommender systems are useful in situations where the resources are limited in number, and the intention is to give access to such resources to users of roles where they are actually requested.

Punjabi University, Patiala, INDIA

Access Control in Cloud Computing

The primary concern of access control is to provide security. In situations where resources are limited and/or maximum utilization of resources is desired, such as in an Enterprise cloud, access control can be used to limit resource access to only those users who require it. However, this need can vary over time. In traditional access control models, the challenge is that access control policies are static, and there is no recommender system to modify them based on resource usage. This paper proposes recommendation algorithms to address this issue. The paper [2] aims to

present an understanding and a brief discussion of research into cloud computing access control technology. The survey of access control models and policies [3], especially related to cloud computing, examines the different models and technologies and analyzes the benefits and drawbacks of each model. [4] reviewed the challenges concerning authentication, access management, security, and services in the cloud environment and proposed solutions to address these issues. [5] described numerous access control mechanisms in the cloud computing environment for various purposes. The novelty of this paper is proposing the use of access control to improve the overall service of an enterprise cloud with the goal of achieving maximum utilization with the least number of resources. This aspect has not been addressed by other suggested improvements in access policies.

2. Methodology

The lack of flexibility to support a variety of users and resources in a dynamic and heterogeneous environment needs a self-adaptive dynamic access control model [6][7]. Such dynamism and adaptiveness for access control is introduced in this paper. The idea of context-aware decision-making and adaptability to all situations [8] is incorporated in the paper. The notion of a recommender system [9] to adapt the system to current needs and use is also a feature of this study. The approach of using clustering to address the interests of individual roles in this paper is derived from the study [10]. A few-shot learning is used in the presented algorithms, and learning is used to improve the performance of the access control mechanism [11].

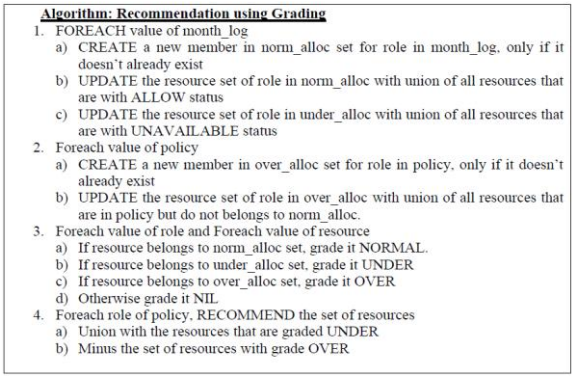
RECOMMENDER ALGORITHMS

The proposed algorithms use user-defined data types that are presented in Figure 1. The REQUEST_STATUS lists the possible values for it. The DISCARDED value is assumed for the request that is discarded due to non-availability of resources, either because of a lack of permission as per the stated policy of the role of the owner of the request or because the limit of permissible instances of allowed resources has been exceeded. The ACCEPTED value is assumed in the case when all requested resources with the desired number of instances are granted by the access policy. The request is in a WAITING state when the Policy Enforcement Manager (PEM) module is in the process of deciding whether to grant the requested resources as per the defined policy for the role of the owner of that request. The PROCESSING is the state of the request when the request is submitted to the cloud, after the ALLOWED state of the request, for further processing by the enterprise cloud. The request remains in this state as long as it is using the cloud. The request is in a COMPLETED state after the request has completed its task on the cloud. The other type, ALLOCATION_GRADE, is for assigning a value as a grade to each resource in every role. The grade is OVER when none of the users of a role has made a request for the resource in the period under study. It is UNDER when a user has made a request for that resource but it is not allowed as per the policy of the role. It is NORMAL when the requested resources are allowed as per the policy of the role. Initially, the grade of all resources is NIL. The RESOURCE_REQUEST_STATUS is ALLOWED for the resource if it is permitted as per the policy, UNAVAILABLE if it is not permitted, and the value is BEYOND_LIMIT if the requested number of instances of the permitted resource exceeds the limit specified in the policy.

```
enum REQUEST_STATUS {DISCARDED, ACCEPTED, WAITING, PROCESSING,
COMPLETED;}
enum ALLOCATION_GRADE {OVER, NORMAL, UNDER, NIL;}
enum RESOURCE_REQUEST_STATUS {ALLOW, BEYOND_LIMIT, UNAVAILABLE;}
```

Figure 1: Enumerated Types

The structure definition of the entities used in algorithm are shown in Figure 2. However, it is mentioned that the name of members of each structure is chosen to reflect its explanation and that the explanation of each structure is beyond the scope of the paper.



Algorithm: Recommendation using Grading

1. FOREACH value of month_log
   a) CREATE a new member in norm_alloc set for role in month_log, only if it doesn't already exist
   b) UPDATE the resource set of role in norm_alloc with union of all resources that are with ALLOW status
   c) UPDATE the resource set of role in under_alloc with union of all resources that are with UNAVAILABLE status
2. Foreach value of policy
   a) CREATE a new member in over_alloc set for role in policy, only if it doesn't already exist
   b) UPDATE the resource set of role in over_alloc with union of all resources that are in policy but do not belongs to norm_alloc.
3. Foreach value of role and Foreach value of resource
   a) If resource belongs to norm_alloc set, grade it NORMAL.
   b) If resource belongs to under_alloc set, grade it UNDER
   c) If resource belongs to over_alloc set, grade it OVER
   d) Otherwise grade it NIL
4. Foreach role of policy, RECOMMEND the set of resources
   a) Union with the resources that are graded UNDER
   b) Minus the set of resources with grade OVER

Figure 2: Structures of data structures

The algorithm, inspired by the idea [12], uses the grading of each resource in every role to recommend a revision of the access policies for each role, as presented in Figure 3. In this algorithm, the log file month_log contains a record of all the requests made during the specified period.

```
struct policy {roles, {resource, instances}};
struct resource_request {resource, instances};
struct request {user_id, req_id, roles, {res_req}, stat};
struct month_log {user_id, roles, req_id, req_stat, {res_req_stat})
struct grade_alloc{role, {resources}}norm_alloc, under_alloc, over_alloc;
struct cluster_set{roles. {resource}};
struct role_res_prob{roles, {resource, prob}};
struct res_role_prob{resource, {roles, prob}};
struct weight_set{ roles, {resource, weight}};
struct role_res_ptage{roles, {resource, ptage}};
```

Figure 3: Grading Algorithm

Clustering [13] means grouping based on some similarity measure. In this context, the individual roles that are in the same cluster are expected to request a similar set of resources among the total available resources because their interests are similar. The algorithm for recommendations using clusters is shown in Figure 4. A two-class cluster is created for each role. The resources that appear in the requests of the users of each role are classified into one class, while the resources that did not appear in the requests are classified into the other class. The recommendations are then incorporated into the policies by either taking the union or the difference of the resources from the existing policy, depending on the class of the resource in the cluster.

Algorithm: Recommendation using Clusters

1. Foreach value of month_log
   a) CREATE a new member in cluster set for role in month_log, only if it doesn't already exist
   b) UPDATE the resource set of role in cluster set with union of all resources requested by the user of that role.
2. Foreach role of policy, RECOMMEND its allowable set of resources
   a) Union with resources that belong to cluster but don't belong to policy
   b) Minus the resources that belong to policy but has no membership in cluster set

Figure 4: Clustering Algorithm

The algorithm devised by the studies [14][15] provides recommendations and prioritizes them, as shown in Figure 5. Prioritization is based on the calculated weight of each resource in every role. The weight calculation procedure involves two main steps: first, finding the probability of resources being requested by users of the role, and second, finding the probability that the user of the role has the resource in their request. These probabilities are calculated based on the log of requests made during the observation period. The weight is calculated as the product of both probabilities.

Algorithm: Recommendation using Weights

1. Foreach value of month_log
   a) Create a new member in role_res_prob set for role in month_log, only if it doesn't already exist
   b) Calculate the number of occurences as COUNT of each resource in all requests for every role
   c) Calculate the total number of resources as TOTAL_COUNT in all requests for every role
   d) COUNT and TOTAL_COUNT gives the probability that a given role has this resource
2. Foreach value of Resource and for it Foreach value of month_log
   a) Create a new member in res_role_prob set for resource in Resource, only if it doesn't already exist
   b) Calculate the number of occurrences as COUNT of a resource in all requests for every role
   c) Calculate the total number of occurrences of resource as TOTAL_COUNT in all requests in all role
   d) COUNT and TOTAL_COUNT gives the probability of each resource in a given role
3. For each value of Role and for it Foreach value in Resource
   a) Create a new member in weight_set for role in Role and/or for resource in Resource, only if it doesn't already exist.
   b) For every role and for for each resource calculate weight
   c) Weight = value of role_res_prob * value of res_role_prob for corresponding role and resource
   d) Normalize the weights
   e) Sort the resources in the order of weights
4. Foreach role of policy, RECOMMEND the set of resources
   a) Union with resources that are with permissible weight in weight set but don't belong to policy
   b) Minus the resources that belong to policy but are out of permissible limit in weight set.

Figure 5: Weightage Algorithm

Figure 6 shows a variant of the algorithm described earlier, which gives recommendations with a prioritization factor. In this variant, the ordering of the recommendations is determined by the percentage chance of a resource appearing in requests. The percentage chance is evaluated from the log files.

Algorithm: Recommendation using Percentage of chance

1. Foreach value of month_log
   a) Create a new member in role_res_ptage set for role in month_log, only if it doesn't already exist
   b) Calculate the number of occurences as COUNT of each resource in all requests for every role
   c) Calculate the total number of resources as TOTAL_COUNT in all requests for every role
   d) COUNT and TOTAL_COUNT gives the chances in percentage that a given role has this resource
2. Foreach role of policy, RECOMMEND the set of resources
   a) Union with resources that are with permissible percentage of chances but don't belong to policy
   b) Minus the resources that belong to policy but are out of permissible limit in percentage of chances

Figure 6: Chance Algorithm

3. Results and Discussions

To demonstrate the effectiveness of the algorithms presented above, a simulation was conducted. The results shown before the implementation of the algorithms represent the performance of the traditional static Role-Based Access Control mechanism. Figure 7 displays the grading of resources in each role before and after applying the proposed algorithms. The comparison between the before and after results indicates a considerable increase in resources with normal grading and a decrease in resources with over and under grading, ultimately leading to improved resource utilization.

Figure 7: Resource Grading Before and After Recommendation

The availability of resources for a given request increases for each role, and there is a decrease in the unavailability of resources as shown in Figure 8. As a result, the acceptance ratio of the requests increases, which is one of the objectives of the above-stated algorithm(s).

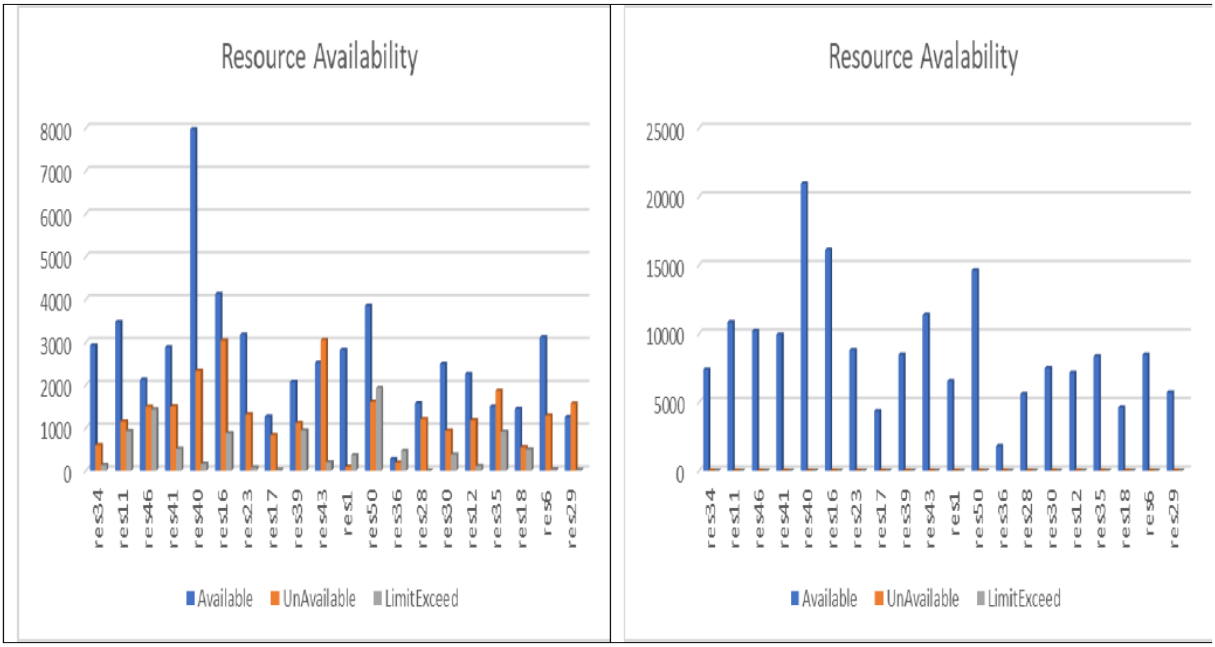


Figure 8: Resource Availability Before and After

The charts are drawn with values in the sample output obtained before and after application of above stated algorithm(s). The desired effect is to

- increase in true positive (improve normal allocation)
- decrease in false negative (reduce under allocation)
- decrease in false positive (reduce over allocation)
- increase in true negative (reduce over allocation)

The performance metric accuracy and precision from confusion matrix are shown in Figure 9. The improvement in both parameters is noticeable. A positive trend of these parameters shows the effectiveness of stated algorithm for true recommendations.

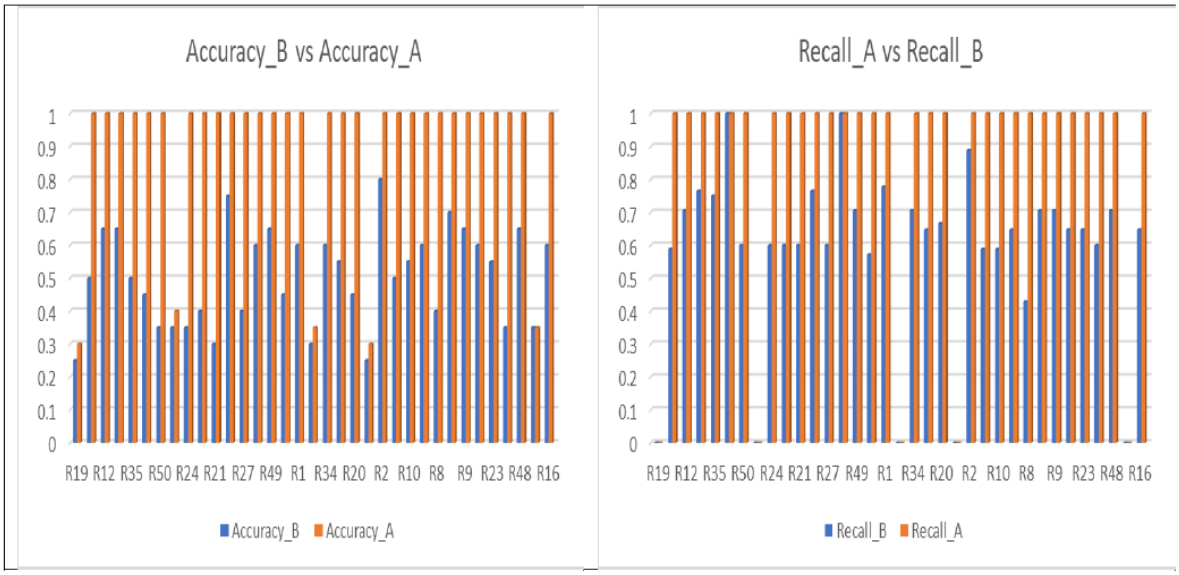


Figure 9: Accuracy and Recall Before and After Recommendation

4. Conclusions

The limitations of traditional Role-Based Access Control (RBAC) policies are in dynamic work environments where user resource needs change over time. RBAC is a common access control model used by organizations to regulate users' access to resources based on their roles and responsibilities within the organization. However, in a dynamic working environment, RBAC policies might not provide sufficient flexibility to adapt to the changing needs of users. Enterprises aim to serve all users with the least possible resources and use access control policies, in addition to their primary concern of security, to restrict the access of resources to those who need them. The study aims to improve resource utilization in the enterprise cloud by adapting RBAC policies to the dynamic requirements of users, with the same set of resources, in completing their tasks. This involves adjusting access policies in real-time to reflect changes in users' roles and responsibilities as they work on their tasks, ensuring that they have access to the resources they need when they need them. To achieve this goal, the study proposes algorithms that can analyze user behavior within each role and suggest appropriate policy adaptations. These algorithms identify patterns in users' behavior and recommend policy changes that optimize resource utilization. The study demonstrated that the recommended algorithms effectively improve resource utilization. Overall, the study highlights the importance of adapting access control policies to the changing needs of users in dynamic work environments with the least number of resources and proposes a solution to address this challenge.

References

[1] Sukesh Bhardwaj & Surendra Yadav, “Role Based Access Control in Clod Computing Security”, International Journal of Scientific & Engineering Research Volume 11, Issue 6, June-2020, pp 784-788

[2] Minghao Wang, “A Survey of Cloud Computing Access Control Technology”, Journal of Physics: Conference Series, Volume 1187, Issue 3, 2019, https://doi.org/10.1088/1742-6596/1187/3/03

[3] Fangbo Cai, Nafei Zhu, Jingsha He, Pengyu Mu, Wenxin Li & Yi Yu, “Survey of access control models and technologies for cloud computing”, Springer Science+Business Media, LLC, part of Springer Nature 2018, Cluster Computing https://doi.org/10.1007/s10586-018-1850-7

[4] Indu, I., Anand, P. M. R., & Bhaskar, V., "Identity and access management in cloud environment: Mechanisms and challenges," International Journal on Engineering Science and Technology, vol. 21, no. 4, pp. 574-588, 2018.

[5] Karataş, G., & Akbulut, A., "Survey on access control mechanisms in cloud computing.," *Journal of Cyber Security and Mobility,* pp. 1-36, 2018.

[6] Hongfa Ding, Changgen Peng, Youliang Tian, and Shuwen Xiang, “A risk adaptive access control model based on Markov for big data in the cloud”, *International Journal of High Performance Computing and Networkingm* 2019 13:4, 464-475. https://doi.org/10.1504/IJHPCN.2019.099269

[7] ALAmri & S. M. S. . An Intelligent Access Control Model. In P. Li, P. A. R. Pereira, & H. Navas (Eds.), Quality Control - Intelligent Manufacturing, Robust Design and Charts. IntechOpen.2021, https://doi.org/10.5772/intechopen.95459

[8] Miguel Calvo, Marta Beltrán, “A Model For risk-Based adaptive security controls”, *Computers & Security*,Volume 115,2022, https://doi.org/10.1016/j.cose.2022.102612.

[9] Polignano, M. and Semeraro, G. Special Issue on Information Retrieval, Recommender Systems and Adaptive Systems. Information 2022, 13, 457. https://doi.org/10.3390/info13100457

[10]Yang, Yz., Zhong, Y. & Woźniak, M., “Improvement of Adaptive Learning Service Recommendation Algorithm Based on Big Data”, *Mobile Netw Appl* **26**, 2176–2187 (2021). https://doi.org/10.1007/s11036-021-01772-y

[11] Aodi Liu, Xuehui Du & Na Wang., “Efficient Access Control Permission Decision Engine Based on Machine Learning”, *Security and Communication Networks*. 2021. https://doi.org/10.1155/2021/3970485

[12] Madni, Hamid & Shafie, Abd Latiff & Yahaya, Coulibaly & Abdulhamid, Shafi’i. (2017). Recent advancements in resource allocation techniques for cloud computing environment: A systematic review. Cluster Computing. 20. 1-45. 10.1007/s10586-016-0684-4.

[13] Saquib Sarfraz, Vivek Sharma, Rainer Stiefelhagen, “Efficient Parameter free Clustering using first neighbor relations”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 8934-8943

[14] Bangweon Song, Seokjoong Kang, "A Method of Assigning Weights Using a Ranking and Nonhierarchy Comparison", *Advances in Decision Sciences*, vol. 2016, Article ID 8963214, 9 pages, 2016. https://doi.org/10.1155/2016/8963214

[15] Ezell, Barry, Christopher J. Lynch, and Patrick T. Hester. 2021. "Methods for Weighting Decisions to Assist Modelers and Decision Analysts: A Review of Ratio Assignment and Approximate Techniques" *Applied Sciences* 11, no. 21: 10397. https://doi.org/10.3390/app112110397

Amardeep Kaur is a faculty member in the subject of Computer Science at Punjabi University Centre for Emerging and Innovative Technology, Mohali. She has done her B.E. (CSE) from Thapar University, Patiala and M.E.(CSE) from Panjab Engineering College, Chandigarh. She is currently pursuing her Ph.D. in Computer Engineering. The area of research is the development of mechanism of access control for effective resource allocation in enterprise cloud. She is credited with various publications in journals and as well in the proceedings of international and national conferences. She has guided a number of M.Tech (CSE) students for their research dissertations.