Hindawi
Journal of Electrical and Computer Engineering
Volume 2023, Article ID 3922393, 30 pages
https://doi.org/10.1155/2023/3922393

Research Article

Adaptive Access Control Mechanism (AACM) for Enterprise Cloud Computing

Amardeep Kaur and Amandeep Verma

Punjabi University, Patiala, India

Correspondence should be addressed to Amandeep Verma; vaman71@gmail.com

Received 6 April 2023; Revised 31 May 2023; Accepted 1 July 2023; Published 13 July 2023

Academic Editor: Ayman Al-Quraan

Enterprise cloud computing provides various services to enterprises, but access to these services is controlled by a firewall. The firewall determines the actions and operations a legitimate user can perform on the available resources. Access control policies allow or restrict access to resources, and they also keep a record of attempted access. In the role-based access control model, access to resources is based on a user's role in the enterprise. As resources are limited, the policy manager has to create policies that optimize resource availability to different roles to improve overall resource utilization. However, this optimization is challenging without prior knowledge of user behaviour and resource requirements for each role. Due to insufficient knowledge, some resources may be available to the wrong roles, while others may be required by other roles but are inaccessible. This results in decreased resource utilization, requiring the redefinition of access control policies with optimal resource availability. The optimal allocation of resources can be achieved by analyzing user behaviour under different roles. The study proposes a novel method for access control that utilizes role profiling and redefines access control policies for different roles to optimize resource availability. Formal methods are employed to ensure accurate system behaviour in software and hardware systems. Formal specifications provide a high-level representation of system behaviour and characteristics. This paper proposes formal specifications using the "*Z*" language to ensure accurate system behaviour in access control mechanisms. The proposed mechanism is implemented in a simulated environment and validated using four variants of the recommender approach. The study concludes that the proposed mechanism consistently enhances operational capability, minimizing over- and under-allocation of resources to roles and improving overall resource utilization within the enterprise. The proposed method is beneficial in dynamic environments where the system must adapt to evolving scenarios.

1. Introduction

Enterprise cloud computing offers infrastructure, software, and platform services to an enterprise whose access is controlled by a firewall. Enterprise cloud computing results in better speed and performance of computing resources as well as improved utilization and lower operational and infrastructure costs for an enterprise. Enterprise cloud computing provides a secure computing environment, providing the capability for access control policies, where decisions are based on numerous factors, such as the role of the user defined within the enterprise, the type of data or application being accessed by the user, and the kind of device being used. Enterprise clouds are better than conventional on-premise servers and other storage systems because they are faster, reduce latency, and prevent data loss.

The primary objective of access control is to ensure security. In situations where resources are limited or optimizing their utilization is important, such as in an enterprise cloud environment, access control can be employed to restrict resource access to only those users who genuinely require it. However, this requirement may change over time. Traditional access control models face a challenge in that their policies are static and lack a recommender system to adapt them based on resource usage. This paper introduces a mechanism to tackle this issue. The novelty of this paper lies in proposing the utilization of access control to enhance the overall service of an enterprise cloud, aiming to achieve maximum resource utilization with a minimal number of resources used. Other suggested improvements in access models in literature have not addressed this particular aspect. In the context of present paper, adaptiveness refers to the ability of the access control model to dynamically modify its policies based on the changing requirements of different roles within an enterprise. It recognizes that the access requirements of users may vary over time and aims to provide a flexible and responsive approach to managing resource access.

In the context of the present study, an enterprise has users and resources. The users of the enterprise are intended to use the resources of the enterprise to accomplish their tasks. The resources are not physically and dedicatedly allocated to any of the users. The resources are provided logically to the users based on their needs, specified in their requests, but as per the policies of the enterprise that are defined as access control policies of the resource, depending on the role of the user requesting the resource(s). The resources are available in the form of an enterprise cloud. Roles are assigned to every user depending on their assigned duties and responsibilities. In order to use resources of the cloud, a user has to submit a request to the cloud through the access control policy module. The policy manager of the access control module either accepts or discards the request based on the request submitted by the user and policies defined for the role of the user. The accepted request is forwarded to the cloud for processing. At regular intervals of time, the log entries that encompass the status of the user requests of each role are analyzed, and a number of reports are generated. The requests are analyzed with the intention to define the role profile in terms of resources because of the variation in user requests for resources in every role. The purpose of the analysis is to identify over-availability and under-availability of the resources as per the policies defined by the enterprise for each role. The analysis of log entries regarding the resource request from a user, request, and role perspectives helps in the redefinition of access control policies. The intention behind the redefinition is to increase the overall resource utilization and optimal availability of the resources of the enterprise to their users.

The paper includes a review of access control, its policies, and models in Section 2. This section also contains a review of access control in cloud computing. Section 3 describes the formal specifications in the "Z" language and behaviour of the proposed adaptive access control mechanism. Section 4 provides a performance evaluation of the proposed study using four variants to suggest recommendations and redefinition of policies to avoid any over- and under-utilization of resources at that time. Section 5 concludes the study. The intended audience for the paper is researchers interested in improving the utilization of computing resources of an enterprise cloud by controlling access control policies.

2. Review of Literature

For the present study, we have conducted a thorough review of various papers related to access control in general and specifically for cloud computing. These papers have been selected based on their relevance and significance to our research topic. The review has provided us with a comprehensive understanding of the current state-of-the-art techniques and approaches used for access control in cloud computing. We have also identified the gaps in the existing literature and the research opportunities that can be explored to improve the access control mechanisms in enterprise clouds. The insights gained from this review have been used to develop our proposed adaptive access control mechanism and to evaluate its performance against existing methods.

2.1. Access Control. An important security aspect of an organization is to safeguard its data and resources for unauthorized revelation or modifications [1–3]. Access control covers three functions: authentication, authorization, and accountability [4]. Authentication is the process of verifying the identity of a user who requests access to a resource. This process involves the submission of a user ID and a password or other credentials to prove the user's identity. Authorization is the process of determining whether a user is allowed to access a specific resource or perform a particular operation on that resource. This process involves comparing the user's credentials to the access control rules defined for the resource. Accountability is the process of tracking the actions of users who access resources and recording them in a log. This information can be used to trace security breaches or violations of access control policies.

The development of an access control system demands that the rules to control access be defined. It follows a multiphase process based on the following concepts [1]:

Security policy: it defines the (high-level) rules according to which access control must be regulated.

Security model: It provides a formal representation of the access control security policy and its functioning. The formalization allows the proof of properties on the security provided by the access control system being designed.

Security mechanism: it defines the low-level (software and hardware) functions that implement the control imposed by the policy and formally stated in the model.

In the context of cloud computing, access control plays a crucial role in ensuring the security of the cloud environment. Cloud computing provides a shared computing environment, which means that multiple users and applications share the same physical infrastructure. As a result, the access control policies must be carefully designed and implemented to prevent unauthorized access to sensitive data and resources. Cloud providers typically offer various access control mechanisms and tools that can be used to manage user access to cloud resources.

2.2. Access Control Policies and Models. Access control is a fundamental concept in computer security that refers to the process of selectively restricting access to resources or data within a computing environment. Access control models provide a framework for enforcing access control policies that define who can access what resources and under what conditions.

The initial access control models focused on identity-based access control, where access is granted based on the identity of the user. The initial work in this direction was proposed by Lampson [5]. The proposed model uses the access control matrix as framework for reasoning about the permitted access in the computing environment. In 1973, Bell-LaPadula [6] proposed a model of protection systems that deals with the control of information flow and assigns access permissions to users based on specific rules. Sandhu and Samarati [2] discussed various access control policies, and the authors also suggested the role-based access control to be an appealing substitute to conventional access control. However, as computing environments became more complex, new access control models were proposed to address the limitations of identity-based access control.

The three main categories of access control models [1, 7–9] are discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC). DAC policies allow users to control access to resources based on their discretion. MAC policies, on the other hand, are centrally controlled and enforce mandatory rules for access control. RBAC policies [9] control access based on the roles assigned to users within the system. Hu et al. [3] explained some of the commonly employed access control services offered in IT systems.

RBAC is considered to be the most promising access control policy for complex environments, as it provides a flexible and scalable approach [10] to access control that can be easily customized to meet the specific needs of different organizations.

Among these policies, RBAC is the most promising access control policy for complex environments [11].

Other access control policies have also been proposed, including attribute-based access control [12–14], gateway-based access control [15], and context-aware access control [4, 16, 17]. These policies are designed to address specific access control requirements in different computing environments. The paper [18] is to introduce and delve into the concept of activity-centric access control (ACAC) within smart and connected ecosystems, emphasizing its significance, vision, and research agenda. The study [19] introduces a novel approach to data security using multiple device shadows (digital twins) to separate and restrict access to data points based on assigned tags. The work [20] proposes ReBAC IoT, an attribute-aware relationship-based access control model for smart IoT systems, considering social relationships among users and attributes to enable dynamic and fine-grained access control. The paper [21] focuses on examining the problem of user attribute reachability, specifically considering attributes assigned directly to users as well as those inherited through group memberships. The paper [22] introduces a hybrid RBAC model that combines offline deep reinforcement learning and Bayesian belief networks to dynamically improve the initial RBAC policy while ensuring compliance with system security rules.

2.3. Access Control in Cloud Computing. Neumann [23] gave the insight into the barriers and its fixes to offer a reliable cloud computing environment. Khan [12] presented various access control methods used in cloud computing and highlighted attribute-based access control features required in dynamic environments. Meghanathan [7] reviewed various dynamic access control models to achieve cross-domain authentication. Li and Zhao [24] highlighted the key challenges towards access control in cloud computing environment and the ensuing research directions. Ahmed and Ashraf Hossain [25] presented a study of the cloud computing concepts involving security issues with respect to cloud computing and cloud infrastructure. Majumder et al. [26] discussed the issues and challenges faced in the cloud computing environment. Aluvalu and Muddana [27] presented the analysis of different access control models for cloud computing elucidating the problems faced by these models and their viable solutions. Indu et al. [28] reviewed the challenges concerning authentication, access management, security, and services in cloud environment and proposed the solutions to address these issues. Various existing access control schemes were examined for the security concerns in cloud computing environment [29]. Karataş and Akbulut [30] described numerous access control mechanisms in cloud computing environment for various purposes. El Sibai et al. [31] presented a survey on access control mechanisms for cloud computing.

Cha et al. [32] presented an attribute-based access control (ABAC) model that controls access based on the attributes of requestors, services, resources, and environment. Lin et al. [33] provided a trust-based access control mechanism for cloud computing by incorporating the trust into the cloud environment. Younis et al. [8] proposed an access control model for cloud computing AC3 to meet the identified cloud access control requirements. It ensured the secure sharing of resources, providing diverse access control to the same cloud user, and gave user the capability to utilize various services securely. The researchers suggested different types of access control models for the cloud computing environments [34]. To address the vulnerability in data privacy of cloud users in cloud computing environment, a fine-grained access control mechanism using cryptography was proposed. The paper [35] covers several features of access control mechanisms, including attribute-based encryption, role-based and hierarchical identity management, identity-based authentication, and trust-based models that are suitable for cloud computing environments. A new enhanced authorization approach to access control in cloud was proposed [36]. The study [37] presents an online malware detection system that utilizes process level performance metrics, evaluating the efficacy of various baseline machine learning models. A smart healthcare cloud using smart contract access control [38] was introduced. A role-based access control for cloud storage [39] was proposed. The paper [40] evaluates and analyzes access control mechanisms in cloud computing based on centralized and decentralized models and presents a comparison of each model's advantages and limitations and discusses the challenges associated with access control in the cloud.

Overall, the studies mentioned focus on the challenges and solutions related to access control in cloud computing environments. These challenges include issues with authentication, access management, security, and privacy. Various access control models are discussed such as attribute-based access control (ABAC) [41], trust-based access control, and fine-grained access control using cryptography. The studies highlight the importance of developing reliable and effective access control mechanisms to ensure the reliable sharing of resources and services in cloud computing environments.

3. Design of the Adaptive Access Control Mechanism

The mechanism presented in the paper introduces a fresh approach to access control mechanisms that can enhance the efficient utilization of cloud resources. This approach involves utilizing role profiling and adjusting access control policies for different roles to optimize the availability of cloud resources while maintaining security. The mechanism is especially beneficial in scenarios where access control policies can also be leveraged to enhance resource utilization, leading to better overall efficiency. By implementing this mechanism, enterprises can streamline their resource allocation processes, improve the response time to requests, and ensure optimal utilization of their cloud resources, thereby improving operational efficiency and reducing costs. The implementation cost of the proposed approach involves efforts in creating and maintaining log files to capture user requests, analyzing the log file contents using a recommender algorithm, and making access control decisions based on the recommendations generated. It includes resources for developing and managing the log files, designing and training the recommender algorithm, integrating it into the system, and implementing decision-making logic. Additionally, integration, deployment, maintenance, monitoring, training, and documentation efforts are also necessary.

3.1. Requirements of the Mechanism. The basic requirements of the stated mechanism are as follows:

- (i) A user is identified by a unique ID and designation
- (ii) A user may have additional charges
- (iii) Every user has at least one role
- (iv) Designation and additional charges may map to different roles
- (v) There are a number of users with same role
- (vi) Each role has the availability of subset of the available resources
- (vii) Available resources are allocated among the various roles in a controlled manner
- (viii) A user may have a number of requests
- (ix) A request is identified by the unique ID and the information related to the owner of the request
- (x) A request belongs to single user under one role
- (xi) A request is either accepted or discarded on the basis of the defined policies
- (xii) Policies are defined in terms of instances available for a given resource for every role

3.2. Features of the Mechanism. The proposed access control mechanism introduces several additional features to the traditional RBAC model to enhance its capabilities in improving resource utilization in dynamic environments.

Firstly, the mechanism is designed to be dynamic and adaptable to changing operational scenarios. This means that the access control policies can be redefined at regular intervals to optimize resource allocation based on changing demand and availability.

Secondly, the mechanism allows a user to have multiple roles, but the applicable policies are determined by the role requested. The user has one designated role, while additional roles may be assigned with additional charges. This enables more flexible resource allocation based on the specific needs of each role.

Thirdly, the mechanism supports requests for multiple resources with multiple instances in a single request. The access policy includes limits on the number of instances for each allowed resource, as well as a limit on the number of instances for each role. This ensures that resources are allocated optimally and efficiently, while also preventing over-allocation and under-utilization of resources.

Finally, the mechanism provides results that aid in decision making about the addition, deletion, and redistribution of resources among roles to optimize utilization at a lower cost. The mechanism generates reports that can be used to identify over- and under-utilized resources and to identify trends in resource usage, enabling enterprises to make informed decisions about resource allocation.

In summary, the proposed access control mechanism introduces new features that allow for more efficient resource allocation in dynamic environments, ultimately leading to optimized resource utilization. The mechanism achieves this by allowing for dynamic adaptation to changing operational scenarios, allowing users to have multiple roles and requesting multiple resources with multiple instances in a single request, and aiding in the decision-making process for resource allocation. The overall result is increased resource availability with fewer resources, all while maintaining security.

3.3. Formal Specifications and Behaviour of the AACM. Formal methods [42, 43] are procedures based on mathematical models for the design of software and hardware systems. Unlike other design systems, formal methods use mathematical proof as a complement to system testing to ensure correct behaviour. The strength of formal methods lies in their ability to verify the entire state space of the system, and the properties that can be proved to hold in the system will hold for all possible inputs.

Formal specifications provide a concise description of the high-level behaviour and properties of a system. These specifications can be model-oriented by constructing a model of the system behaviour using mathematical objects such as sets and sequences. Alternatively, they can use a set of necessary properties to describe system behaviour, such as axioms and rules. Formal specifications provide several benefits, including

(i) Higher level of rigor: Formal specifications offer a higher level of rigor, which leads to better problem understanding. This helps ensure that the system meets its requirements and specifications.

(ii) Uncovering defects: Formal specifications help uncover defects that may be missed when using traditional specification methods. By using a formal notation, it becomes easier to identify problems and inconsistencies in the system design.

(iii) Early defect detection: Formal specifications allow for early defect detection, which can save time and reduce costs. By detecting defects early in the development process, it is easier to fix them before they become major problems.

(iv) Self-consistency: The semantics of formal specification languages allow for verification of self-consistency. This means that the specification itself can be checked for consistency, which helps ensure that the system will behave as intended.

(v) Formal proofs: Formal specifications facilitate the use of formal proofs to establish fundamental system properties and invariants. This can help ensure that the system meets its requirements and specifications.

Overall, formal specifications provide several benefits that can help ensure the correctness and reliability of software and hardware systems.

Many formal notations have been developed for writing formal specifications.

The *B* method [44] is a method of software development based on *B*, a tool-supported formal method based on an abstract machine notation, used in the development of computer software. It was originally developed by Jean-Raymond Abrial. *B* is related to the *Z* notation.

VDM [45] was developed at the IBM laboratories in Vienna. The current version of the VDM specification language, VDM-SL, has been standardized by the International Standards Organization (ISO). It supports the modelling and analysis of software systems at different levels of abstraction. Using VDM-SL constructs, both data and algorithmic abstractions expressed in one level can be refined to a lower level to derive a concrete model that is closer to the final implementation of the system.

Z is a formal specification language [46] based on Zermelo set theory. It was developed at the Programming Research Group at Oxford University in the early 1980s and became an ISO standard in 2002. *Z* specifications are mathematical and employ a classical two-valued logic. The use of mathematics ensures precision and allows inconsistencies and gaps in the specification to be identified. Theorem provers may be employed to demonstrate that the software implementation meets its specification [47].

Z has been used to underpin a model of RBAC [48]. The *Z* specification is created for the commercial application of online food ordering system to improve the order detail accuracy and efficiency [49]. *Z* specification language is used to design the e-commerce system and specify security constraints [50]. A scanner and parser for *Z* specifications [51] was introduced.

Other formal specification languages include

(i) CSP (communicating sequential processes) [52]: a formal language for describing patterns of interaction in concurrent systems.

(ii) TLA+ (temporal logic of actions): a language for specifying and verifying concurrent and distributed systems, developed by Leslie Lamport.

(iii) Alloy [53]: a lightweight specification language and analyzer for software modelling and analysis, developed at MIT.

Each of these languages [54] has its own strengths and weaknesses, and the choice of language depends on the specific requirements and constraints of the system being modelled.

3.3.1. Formal Specifications.

3.3.1. Formal Specifications. In the study, the specification pattern is utilized to define the formal specifications of the proposed mechanism. This pattern involves employing the syntax and semantics of the *Z* language to capture the desired behaviour and properties of the system or process being modelled.

The enterprise is defined as $E = \{\text{USERS, RESOURCE}\}$ where USERS is the set of users and RESOURCES is the set of available computing resources. So, USERS = $\{u1, u2, u3, \ldots, un\}$ is a finite set of users where each user is identified by a set of attributes. The USER_ID is to uniquely identify the user, USER_DESIG is attributed to designation of the user, a finite set of ROLES is assigned to user on the basis of their designation and additional charges assigned, ADDL_-CHARGE is a finite set of additional charges held by the user, and ALLOCATED_RESOURCES is a finite set of tuples of resource and the number of instances of that resource allocated to the user at a given instant for each of the assigned role.

RESOURCES = $\{\text{res}1, \text{res}2, \text{res}3, \ldots, \text{res}n\}$ is a finite set of resources. Each resource has one or more instances of each resource.

ROLES = $\{r1, r2, r3, \ldots, rn\}$ is a finite set of roles that are assigned to users.

The statement of the problem is to find the optimized mapping of the members of the RESOURCES set, as resources, to members of the ROLES set. The optimization is with the intention to enhance the resource utilization. Role profiling is used for it. The formal specifications for the USERS, RESOURCES, and ROLES are given in Figure 1.

$[RESOURCE, ROLES]$
$ROLE == ROLES \times ROLE\_USER\_LIMIT \times ROLE\_CURRENT\_USERS$
$[USER\_DESIG, ADDL\_CHARGE]$
$ALLOCATED\_RESOURCES == RESOURCE \times INSTANCES \times ROLES$
$USER == USER\_ID \times USER\_DESIG \times \mathbb{F}\ ROLES \times \mathbb{F}\ ADDL\_CHARGE \times \mathbb{F}\ ALLOCATED\_RESOURCES$

Figure 1: Specifications of users, roles, and resources.

The specifications for various operations on USER set like to add new user and delete an existing one are shown in Figure 2. In order to add a new user, user designation and additional charges (if any) are provided as input. The USER_ID is automatically generated by using a global variable USER_COUNT.

The specifications for mapping User_Desig to ROLES and Addl_Charges to ROLES are shown in Figure 3 along with the operations for adding and deleting a role for a given designation or charge and updating an existing mapping from designation to role or charge to role.

The roles are assigned to users based on their designation and additional charges (if any) held by them. This mapping is shown by the specifications shown in Figure 4.

The specifications of the queries that can be asked on the basis of role assignment like to find the roles for a given designation/charge or vice versa are shown in Figure 5.

The policy database as shown in Figure 6 consists of set of policies POLICY = $\{P1, P2, P3, \ldots, Pn\}$ where each policy is characterized by a tuple of ROLES and RESOURCE_ALLOWED. Here, RESOURCE_ALLOWED itself is a tuple of RESOURCE and INSTANCES indicating the instances of each resource allowed to a particular role.

The specifications to manage the policy database are given in Figure 7.

The specification regarding the request for the resources is specified in Figure 8.

There are some declarations that are required for further operations. The motive behind creating these declarations is to create a log set LOG = $\{\log_1, \log_2, \log_3, \cdots, \log_n\}$ as specified in Figure 9. Here, each log entry is attributed by the identity of the user, the role among the possible roles assigned to that user, and a unique identity of the request. In addition, the status of the request that is initially waiting and completed on successful completion of that request and a set to indicate the status of each requested resources are also recorded. The purpose of the log entries is for further analysis of the requests by the user for optimizing the usage of the resources. LOG is used to record data on day-to-day basis. For analysis on data for more days, MONTH_LOG is maintained. It has the same members as LOG. It appends the daily entries of LOG to it.

The specification regarding the limit on number of users and the number of current users for each role is shown in Figure 10. Here, the limit indicates the maximum allowed users for a given role and current users indicate the number of users for a given role at a given instance of time.

With an aim to characterize the role, a log that encapsulate the details regarding every resource with the number of instances currently allocated, number of times the limit of permissible instances gets exceeded for the given request, and an entry to record the number of moments the requested resource is not available for a given role is specified as shown in Figure 11, . The availability grade is also assigned as per the values in other attributes of the log.

The specifications of the module that makes a decision either to accept or discard the submitted request on the basis of the policies defined as per the credentials of the request, i.e., role of the user in the present context, are defined in Figure 12. The requested resources with their instances are compared to the resources allowed for a given role. The currently allocated resources are also considered for comparison purposes. If the requested resources and the required instances satisfy the criteria specified in the policy, the user request is marked as accepted. Otherwise, the request is graded as discarded. Furthermore, the status of each requested resource is graded as ALLOW, BEYOND_LIMIT, and UNAVAILABLE depending upon the requested resource and specified policies.

The request that is accepted by the access control module is forwarded to the cloud for further processing. The status of the request is changed to processing. The resources available in cloud serve the incoming request. The specifications of the above said functionality are shown in Figure 13.

On the completion of the service to the request, the request status is changed to completed. Furthermore, the values in allocated resources in the user profile are also modified, and it reflects that the resources of the cloud are not more with the user of the request. The specifications are shown in Figure 14.

The specification to query the log for a given role and to query the log for a given user is shown in Figure 15. These types of queries are helpful in the analysis of the requests from the role and from the user perspectives.

The role log is modified by querying the log for requests. The specification of the said module is shown in Figure 16. This modification is an important step as on its basis further recommendations take place for optimizing the availability of resources among the various roles.

On the basis of it, resource availability of each resource for a role is graded. The grade is OVER, NORMAL, or UNDER. The resource is graded as NORMAL, if that resource is in the requests of the users of that role. It is graded as UNDER, if the resource is in the requests of the users of role under observation but it is not available as per the present policy of that role. Resource is with grade OVER, if it is in the policy of the role but it was never requested by the users of that role. The specifications for the same are shown in Figure 17. The stated figure shows the specifications of the module for the creation of OVER_ALLOC, NORM_ALLOC, and UNDER_ALLOC sets of resources for each role by analysis of the MONTH_LOG. The specifications of grading resources using these sets are also shown in this figure.

USERS

$USER: \mathbb{P}\ USER$

ADD_USER

$\Delta USERS$

$desig?: USER\_DESIG$

$charge?: \mathbb{F}\ ADDL\_CHARGE$

$new\_user: USER$

---

if $TOTAL\_USER < LIMIT\_TOTAL\_USER$

then $new\_user\ .\ 1\ .\ uid = USER\_COUNT + 1 \wedge new\_user\ .\ 2 = desig? \wedge new\_user\ .\ 4 = charge?$

$\quad \wedge\ new\_user\ .\ 5 = \{\} \Rightarrow USER' = USER \cup \{new\_user\} \wedge TOTAL\_USER = TOTAL\_USER + 1$

else $USER' = USER$

DELETE_USER

$\Delta USERS$

$userid?: USER\_ID$

$\exists u: USER \mid u \in USER \wedge u\ .\ 1 = userid?$

$\quad \bullet\ USER' = USER \setminus \{u\} \wedge TOTAL\_USER = TOTAL\_USER - 1$

Figure 2: Specifications of operations on user.

ROLE_MAP_DESIG

$role\_map\_desig: USER\_DESIG \rightarrowtail ROLES$

ROLE_MAP_CHARGE

$role\_map\_charge: ADDL\_CHARGE \rightarrowtail ROLES$

AddRoleforDesig

$\Delta ROLE\_MAP\_DESIG$

$new\_desig?: USER\_DESIG$

$new\_role?: ROLES$

$role\_map\_desig' = role\_map\_desig \cup \{(new\_desig? \mapsto new\_role?)\}$

AddRoleforCharge

$\Delta ROLE\_MAP\_CHARGE$

$new\_charge?: ADDL\_CHARGE$

$new\_role?: ROLES$

$role\_map\_charge' = role\_map\_charge \cup \{(new\_charge? \mapsto new\_role?)\}$

RemoveDesig

$\Delta ROLE\_MAP\_DESIG$

$desig?: USER\_DESIG$

$role\_map\_desig = \{desig?\} \lhd role\_map\_desig$

RemoveCharge

$\Delta ROLE\_MAP\_CHARGE$

$charge?: ADDL\_CHARGE$

$role\_map\_charge = \{charge?\} \lhd role\_map\_charge$

UpdateRoleProfileDesig

$\Delta ROLE\_MAP\_DESIG$

$desig?: USER\_DESIG$

$role?: ROLES$

$role\_map\_desig' = role\_map\_desig \oplus \{(desig? \mapsto role?)\}$

UpdateRoleProfileCharge

$\Delta ROLE\_MAP\_CHARGE$

$charge?: ADDL\_CHARGE$

$role?: ROLES$

$role\_map\_charge' = role\_map\_charge \oplus \{(charge? \mapsto role?)\}$

Figure 3: Specifications of role mapping and operations.

```
┌─ROLE_ASSIGNMENT──────────────────────────────
│ ΞROLE_MAP_DESIG
│ ΞROLE_MAP_CHARGE
│ ΔUSERS
│ idesig?: USER_DESIG
│ iuid?: USER_ID
│ charge?: 𝔽 ADDL_CHARGE
│ assigned_role: ROLE
├──────────────────
│ ∃u: USER | u ∈ USER ∧ u . 1 = iuid?
│    • ∀charges: ADDL_CHARGE | charges ∈ charge?
│        • if ∃role: ROLE | {role . 1} = { r: ROLES | charges ↦ r ∈ role_map_charge }
│                  ∨ {role . 1} = { r: ROLES | idesig? ↦ r ∈ role_map_desig }
│                    ∧ role . 3 . current_users < role . 2 . user_limit • true ∧ assigned_role = role
│          then u . 3 = u . 3 ∪ {assigned_role . 1}
│              ∧ assigned_role . 3 . current_users = assigned_role . 3 . current_users + 1
│          else u . 3 = u . 3
└──────────────────────────────────────────────
```

Figure 4: Specification of role assignment.

```
┌─FindRolesforDesig────────────────────────────
│ ΞROLE_MAP_DESIG
│ desig?: USER_DESIG
│ roles!: 𝔽 ROLES
├──────────────────
│ roles! = { r: ROLES | desig? ↦ r ∈ role_map_desig }
└──────────────────────────────────────────────

┌─FindRolesforCharge───────────────────────────
│ ΞROLE_MAP_CHARGE
│ charge?: ADDL_CHARGE
│ roles!: 𝔽 ROLES
├──────────────────
│ roles! = { r: ROLES | charge? ↦ r ∈ role_map_charge }
└──────────────────────────────────────────────

┌─FindDesigforRoles────────────────────────────
│ ΞROLE_MAP_DESIG
│ roles?: ROLES
│ desig!: 𝔽 USER_DESIG
├──────────────────
│ desig! = { d: USER_DESIG | d ↦ roles? ∈ role_map_desig }
└──────────────────────────────────────────────

┌─FindChargeforRoles───────────────────────────
│ ΞROLE_MAP_CHARGE
│ roles?: ROLES
│ charge!: 𝔽 ADDL_CHARGE
├──────────────────
│ charge! = { c: ADDL_CHARGE | c ↦ roles? ∈ role_map_charge }
└──────────────────────────────────────────────

┌─FindRoleOfUser───────────────────────────────
│ ΞUSERS
│ iuid?: ℕ
│ role!: 𝔽 ROLES
├──────────────────
│ ∃u: USER | u . 1 . uid = iuid? • role! = u . 3
└──────────────────────────────────────────────
```

Figure 5: Specification of query role mapping.

```
RESOURCE_ALLOWED == RESOURCE × INSTANCES
POLICY == ROLES × 𝔽 RESOURCE_ALLOWED
┌─POLICY_DATABASE──────────────────────────────
│ POLICY: ℙ POLICY
└──────────────────────────────────────────────
```

Figure 6: Specifications of policies.

```
┌── AddPolicy ──────────────────────────────────────────
│ ΞPolicyDatabase
│ role?: ROLES
│ resource_allowed?: RESOURCE_ALLOWED
│ new_policy: POLICY
│ update_status: POLICY_UPDATE_STATUS
├──────────────────
│ if POLICY_UPDATE ≠ TRUE
│ then POLICY_UPDATE = TRUE
│     ∧ new_policy . 1 = role? ∧ new_policy . 2 = {resource_allowed?} ∪ new_policy . 2
│     ∧ POLICY′ = POLICY ∪ {new_policy} ∧ update_status = DONE ∧ POLICY_UPDATE = FALSE
│ else update_status = PENDING
└───────────────────────────────────────────────────────

┌── FindPolicy ─────────────────────────────────────────
│ ΞPolicyDatabase
│ role?: ROLES
│ policy!: POLICY
├──────────────────
│ ∃p: ROLES × {RESOURCE_ALLOWED} | role? = p . 1 • policy! = p
└───────────────────────────────────────────────────────

┌── UpdatePolicy ───────────────────────────────────────
│ ΔPolicyDatabase
│ role?: ROLES
│ resource_allowed?: RESOURCE_ALLOWED
│ up: POLICY
│ update_status: POLICY_UPDATE_STATUS
├──────────────────
│ if POLICY_UPDATE ≠ TRUE
│ then ∃p: POLICY | p ∈ POLICY ∧ p . 1 = role?
│         • POLICY′ = POLICY \ {p} ∧ up . 1 = role? ∧ up . 2 = {resource_allowed?} ∪ up . 2
│           ∧ POLICY_UPDATE = TRUE ∧ POLICY′ = POLICY ∪ {up} ∧ update_status = DONE
│           ∧ POLICY_UPDATE = FALSE
│ else update_status = PENDING
└───────────────────────────────────────────────────────

┌── DeletePolicy ───────────────────────────────────────
│ ΔPolicyDatabase
│ role?: ROLES
│ update_status: POLICY_UPDATE_STATUS
├──────────────────
│ if POLICY_UPDATE ≠ TRUE
│ then ∃p: POLICY | p ∈ POLICY ∧ p . 1 = role?
│         • POLICY_UPDATE = TRUE ∧ POLICY′ = POLICY \ {p}
│           ∧ update_status = DONE ∧ POLICY_UPDATE = FALSE
│ else update_status = PENDING
└───────────────────────────────────────────────────────
```

Figure 7: Specification to query policy database.

ROLE_RESOURCE_CLUSTER is created to make the clusters of the resources for each role appear in the requests of the users of that role. Here the clusters may overlap, i.e., a resource can be a member of more than one cluster. The module uses the MONTH_LOG to find out the clusters of resources for every role. The specifications of this module are shown in Figure 18.

The specifications of the module to find the weights of every resource in each role are shown in Figures 19 and 20. This module is to take into account the probability of the requests of every resource in each role and the probability of every role having that resource in their requests. Figure 19 shows the specifications of types used in Figure 20.

Figure 20 shows the specification to find ROLE_RES_PROB and RES_ROLE_PROB and then to find the weight of each resource in every role using these probabilities.

The specifications to find the appearance of each resource in the requests for every role in terms of percentage to the total number of requests for that resource in all roles are shown in Figure 21.

(1) Recommenders. The recommender system is responsible for providing recommendations to adjust the access control policies for resources and the number of instances allowed for each role. These recommendations are based on the data collected and organized during the operation of computing

```
RESOURCE_REQUEST == RESOURCE × INSTANCES

REQUEST_STATUS ::= Discarded | Accepted | Waiting | Processing | Completed
REQUEST == USER_ID × REQUEST_ID × ROLES × {RESOURCE_REQUEST} × REQUEST_STATUS
┌─REQUEST_PROFILE──────────────────────────────
│ REQUEST: ℙ REQUEST
└──────────────────────────────────────────────

RESOURCE_REQUEST_STATUS ::= ALLOW | BEYOND_LIMIT | UNAVAILABLE
REQ_RES_STAT == RESOURCE × RESOURCE_REQUEST_STATUS
┌─MAKE_REQUEST─────────────────────────────────
│ ΞREQUEST_PROFILE
│ userid?: USER_ID
│ reqid: REQUEST_ID
│ role?: ROLES
│ res_req?: ℙ RESOURCE_REQUEST
│ req_status: REQUEST_STATUS
│ new_request: REQUEST
├──────────────
│ new_request . 1 = userid?
│ new_request . 2 . rid = REQUEST_COUNT + 1
│ new_request . 3 = role?
│ new_request . 4 = res_req?
│ new_request . 5 = Waiting
│ REQUEST′ = REQUEST ∪ {new_request}
└──────────────────────────────────────────────
```

Figure 8: Specifications for the request.

```
LOG == USER_ID × ROLES × REQUEST_ID × REQUEST_STATUS × ℙ REQ_RES_STAT
┌─LOG_DATA─────────────────────────────────────
│ LOG: ℙ LOG
└──────────────────────────────────────────────

MONTH_LOG == USER_ID × ROLES × REQUEST_ID × REQUEST_STATUS × ℙ REQ_RES_STAT
┌─MONTH_LOG_SET────────────────────────────────
│ MONTH_LOG: ℙ MONTH_LOG
└──────────────────────────────────────────────
```

Figure 9: Specifications for log and month_log.

```
┌─ROLE_USER_LIMIT──────────────────────────────
│ user_limit: ℕ
└──────────────────────────────────────────────

┌─ROLE_CURRENT_USERS───────────────────────────
│ current_users: ℕ
└──────────────────────────────────────────────

ROLE == ROLES × ROLE_USER_LIMIT × ROLE_CURRENT_USERS
┌─ROLE_SET─────────────────────────────────────
│ ROLE: ℙ ROLE
└──────────────────────────────────────────────
```

Figure 10: Specifications of role.

```
┌─LIMIT_EXCEEDED───────────────────────────────
│ limit_exceeded: ℕ
└──────────────────────────────────────────────

┌─NO_AVAILABLE─────────────────────────────────
│ no_available: ℕ
└──────────────────────────────────────────────

ROLE_RESOURCE_LOG ==   RESOURCE × INSTANCES × LIMIT_EXCEEDED × NO_AVAILABLE
ROLE_LOG == ROLES × 𝔽 ROLE_RESOURCE_LOG
┌─ROLE_LOG_SET─────────────────────────────────
│ ROLE_LOG: ℙ ROLE_LOG
└──────────────────────────────────────────────
```

Figure 11: Specification of log for role.

```
┌──REQUEST_SUBMISSION_ACM ──────────────────────────
│ ΞUSERS
│ ΞREQUEST_PROFILE
│ ΞPolicyDatabase
│ ΔLOG_DATA
│ user_request?: REQUEST
│ log: LOG
├──────────────────
│ ∃u: USER | u ∈ USER ∧ u . 1 = user_request? . 1 ∧ {user_request? . 3} ⊆ u . 3
│   • log . 1 = user_request? . 1 ∧ log . 2 = user_request? . 3 ∧ log . 3 = user_request? . 2
│ ∀res_req: RESOURCE_REQUEST; alloc_res: ALLOCATED_RESOURCES
│   | res_req ∈ user_request? . 4 ∧ alloc_res . 3 = user_request? . 3
│   • if ∃pm: POLICY; res_allow: RESOURCE_ALLOWED
│             | res_allow ∈ pm . 2 ∧ pm . 1 = user_request? . 3
│               ∧ res_allow . 1 = res_req . 1 ∧ alloc_res . 1 = res_req . 1
│               ∧ res_allow . 2 . INSTANCE ⩽ res_req . 2 . INSTANCE - alloc_res . 2 . INSTANCE • true
│     then user_request? . 5 = Accepted
│          ∧ log . 4 = Accepted ∧ {log . 5} = {log . 5} ∪ {({res_req . 1} × {ALLOW})}
│     else if ∃pm: POLICY; res_allow: RESOURCE_ALLOWED
│                | res_allow ∈ pm . 2 ∧ pm . 1 = user_request? . 3
│                  ∧ res_allow . 2 . INSTANCE > res_req . 2 . INSTANCE - alloc_res . 2 . INSTANCE
│                • true ∧ user_request? . 5 = Discarded ∧ log . 4 = Discarded
│          then {log . 5} = {log . 5} ∪ {({res_req . 1} × {BEYOND_LIMIT})}
│          else {log . 5} = {log . 5} ∪ {({res_req . 1} × {UNAVAILABLE})}
│ LOG′ = LOG ∪ {log}
└──────────────────────────────────────────────
```

Figure 12: Specifications for request submission to ACM.

```
┌──REQUEST_FORWARDED_CLOUD──────────────────────────
│ ΔREQUEST_PROFILE
│ ΔUSERS
│ reqid?: REQUEST_ID
│
├──────────────────
│ ∃req: REQUEST; usr: USER; rlog: ROLE_LOG
│   | req ∈ REQUEST ∧ req . 2 = reqid? ∧ usr ∈ USER ∧ usr . 1 = req . 1 ∧ rlog . 1 = req . 3
│   • ∀res_req: RESOURCE_REQUEST; res_alloc: ROLE_RESOURCE_LOG;
│       res_alloc_user: ALLOCATED_RESOURCES
│       | res_req ∈ req . 4 ∧ res_req . 1 = res_alloc_user . 1 ∧ usr . 5 = {res_alloc_user}
│         ∧ res_alloc ∈ rlog . 2 ∧ res_req . 1 = res_alloc . 1 ∧ res_alloc_user . 3 = req . 3
│       • res_alloc . 2 . INSTANCE = res_alloc . 2 . INSTANCE + res_req . 2 . INSTANCE
│         ∧ res_alloc_user . 2 . INSTANCE = res_alloc_user . 2 . INSTANCE + res_req . 2 . INSTANCE
│         ∧ req . 5 = Processing ⇒ {rlog . 2} = {rlog . 2} ∪ {({res_req . 1} × {res_alloc . 2} × {} × {} × {})}
│
└──────────────────────────────────────────────
```

Figure 13: Specifications for request forwarded to cloud.

```
┌──REQUEST_COMPLETED_CLOUD──────────────────────────
│ ΔREQUEST_PROFILE
│ ΔUSERS
│ reqid?: REQUEST_ID
├──────────────────
│ ∃req: REQUEST; usr: USER; rlog: ROLE_LOG
│   | req ∈ REQUEST ∧ req . 2 = reqid? ∧ usr ∈ USER ∧ usr . 1 = req . 1 ∧ rlog . 1 = req . 3
│   • ∀res_req: RESOURCE_REQUEST; res_alloc: ROLE_RESOURCE_LOG;
│       res_alloc_user: ALLOCATED_RESOURCES
│       | res_req ∈ req . 4 ∧ res_req . 1 = res_alloc_user . 1 ∧ usr . 5 = {res_alloc_user}
│         ∧ res_alloc_user . 3 = req . 3 ∧ res_alloc ∈ rlog . 2 ∧ res_req . 1 = res_alloc . 1
│       • res_alloc . 2 . INSTANCE = res_alloc . 2 . INSTANCE - res_req . 2 . INSTANCE
│         ∧ res_alloc_user . 2 . INSTANCE = res_alloc_user . 2 . INSTANCE - res_req . 2 . INSTANCE
│         ∧ req . 5 = Completed ⇒ {rlog . 2} = {rlog . 2}∪ {({res_req . 1} × {res_alloc . 2} × {} × {} × {})}
│
└──────────────────────────────────────────────
```

Figure 14: Specification for request completion on cloud.

```
QUERY_LOG_FOR_ROLE_AND_USER
ΞLOG_DATA
role?: ROLES
role_log!: ℙ LOG
userid?: USER_ID
user_log!: ℙ LOG
────────────────
∀lg: LOG | lg . 2 = role? • role_log! = role_log! ∪ {lg}
∀lg: LOG | lg . 1 = userid? • user_log! = user_log! ∪ {lg}
```

Figure 15: Specification for querying log.

```
QUERY_LOG_UPDATE_ROLE_LOG
role?: ROLES
limit_exceeded: ℤ
no_available: ℤ
────────────────
∀lg: LOG | lg . 2 = role? • ∀req_stat: REQ_RES_STAT; res: RESOURCE
    | req_stat ∈ lg . 5 ∧ req_stat . 1 = res ∧ req_stat . 2 = BEYOND_LIMIT
    • ∃rlog: ROLE_LOG; rol_res_log: ROLE_RESOURCE_LOG | rlog . 1 = role? ∧ rol_res_log . 1 = res
            • rol_res_log . 3 . limit_exceeded = rol_res_log . 3 . limit_exceeded + 1
∀lg: LOG | lg . 2 = role? • ∀req_stat: REQ_RES_STAT; res: RESOURCE
    | req_stat ∈ lg . 5 ∧ req_stat . 1 = res ∧ req_stat . 2 = UNAVAILABLE
    • ∃rlog: ROLE_LOG; rol_res_log: ROLE_RESOURCE_LOG | rlog . 1 = role? ∧ rol_res_log . 1 = res
            • rol_res_log . 4 . no_available = rol_res_log . 4 . no_available + 1
```

Figure 16: Specification for role log updation.

services offered by the cloud, to serve the requests made by the users of the cloud. Requests that comply with the access control policies are forwarded to the cloud, while those that do not comply are rejected. The current study proposes four approaches for making these recommendations, and their performance is shown in the next section.

The specifications to provide recommendation and to adapt the policy with the suggested recommendations using grading of the resources are shown in Figure 22. This recommender uses RESOURCE_GRADING_SET that is defined in earlier specifications.

ROLE_RESOURCE_CLUSTER, which is defined above, is used by the module to suggest recommendations and to adapt the policy. The specifications for this module are shown in Figure 23.

The weight of the resource in a role is used to decide either to recommend or reject a resource for every role. This approach is useful where the policy is only adapted to resources in every role where its weight is equal to or more than the threshold weight which is decided by the policy manger. The specifications of this recommender module are shown in Figure 24. The threshold input is input to this module.

In order to adapt the policy with only those resources that appear more frequently in the requests, the percentage recommender is proposed. In this module, the frequency is determined in the form of percentage, a resource requested in a role to the total number of requests with that resource in all roles. The specification is shown in Figure 25. In this module, the percentage threshold is input parameter.

3.3.2. Behaviour of the AACM. A finite state machine to specify the behaviour of the request in terms of its states and the action(s) to change the state is shown in Figure 26.

The state transition diagram of the request is shown in Figure 27.

The state transition table of the request object is shown in Table 1.

The states and action for querying policy and updating it as a finite state machine are shown in Figure 28.

The state transition diagram of the policy database is shown in Figure 29.

The state transition table for policy database is shown in Table 2.

4. Implementation

Using a simulator to generate data and validate the study is a common approach in computer science research, particularly in the field of cloud computing. Simulators are used to create a realistic environment for testing and evaluation, allowing researchers to analyze the performance of proposed mechanisms under various conditions without the need for expensive physical infrastructure. By using a simulator, researchers can generate large amounts of data in a controlled environment and use statistical analysis to draw conclusions about the effectiveness of the proposed mechanism. This can help researchers to identify potential issues and optimize the mechanism before it is implemented in a real-world setting. The proposed mechanism is implemented in a simulated environment to show its effectiveness. The next subsection presents the simulation and the outcomes. The other subsection validates the simulation mechanism.

4.1. Simulation Model and Performance. The simulation model used in the study is presented in Figure 30. The model generates users, roles, and resources based on user inputs.

```
RESOURCE_GRADING == ROLE × RESOURCE × ALLOCATION_GRADE
UNDER_ALLOC == ROLES × 𝔽 RESOURCE
NORMAL_ALLOC == ROLES × ℙ RESOURCE
OVER_ALLOC == ROLES × 𝔽 RESOURCE
MONTH_LOG == USER_ID × ROLES × REQUEST_ID × REQUEST_STATUS × ℙ REQ_RES_STAT
┌─MONTH_LOG_SET──────────────────────────────
│ MONTH_LOG: ℙ MONTH_LOG
└────────────────────────────────────────

┌─OVER_ALLOC_SET─────────────────────────────
│ OVER_ALLOC: ℙ OVER_ALLOC
└────────────────────────────────────────

┌─UNDER_ALLOC_SET────────────────────────────
│ UNDER_ALLOC: ℙ UNDER_ALLOC
└────────────────────────────────────────

┌─NORM_ALLOC_SET─────────────────────────────
│ NORMAL_ALLOC: ℙ NORMAL_ALLOC
└────────────────────────────────────────

RES_GRADE == RESOURCE × ALLOCATION
ROLE_RES_GRADE == ROLES × 𝔽 RES_GRADE
┌─ROLE_RESOURCE_GRADING_SET──────────────────
│ ROLE_RES_GRADE: ℙ ROLE_RES_GRADE
└────────────────────────────────────────

┌─CREATE_UNDER_ALLOC─────────────────────────
│ ΞMONTH_LOG_SET
│ ΔUNDER_ALLOC_SET
│ norm: UNDER_ALLOC
│ res: ℙ RESOURCE
│ stat: ℙ REQ_RES_STAT
├──────────────
│ ∀role: ROLE; month_log: MONTH_LOG | role . 1 = month_log . 2
│    • stat = month_log . 5 ∧ norm . 1 = role . 1 ∧ norm . 2 = res
│ ∀se: REQ_RES_STAT | se ∈ stat ∧ se . 2 = UNAVAILABLE • res = res ∪ {se . 1}
│ UNDER_ALLOC′ = UNDER_ALLOC ∪ {norm}
└────────────────────────────────────────

┌─CREATE_OVER_ALLOC──────────────────────────
│ ΞNORM_ALLOC_SET
│ ΞUNDER_ALLOC_SET
│ ΞPOLICY_DATABASE
│ ΔOVER_ALLOC_SET
│ over: OVER_ALLOC
│ policy: POLICY
│ res: ℙ RESOURCE
│ newres: ℙ RESOURCE
│ res_allow: ℙ RESOURCE_ALLOWED
├──────────────
│ ∀role: ROLE; policy: POLICY | role . 1 = policy . 1
│    • res_allow = policy . 2 ∧ policy = policy
│ ∀ra: RESOURCE_ALLOWED | ra ∈ res_allow • res = res ∪ {ra . 1}
│ ∀ne: NORMAL_ALLOC; ue: UNDER_ALLOC | ne . 1 = policy . 1 ∧ ue . 1 = policy . 1
│    • ∀re: RESOURCE | re ∈ res ∧ re ∉ ne . 2 ∧ re ∉ ue . 2 • newres = newres ∪ {re}
│ over . 1 = policy . 1  ∧ | over . 2 = newres
│ OVER_ALLOC′ = OVER_ALLOC ∪ {over}
└────────────────────────────────────────

┌─ROLE_RESOURCE_GRADING──────────────────────
│ ΞNORM_ALLOC_SET
│ ΞUNDER_ALLOC_SET
│ ΞOVER_ALLOC_SET
│ res_grade: RES_GRADE
│ role_res_grade: ROLE_RES_GRADE
├──────────────
│ ∀role: ROLE | role . 1 ∈ ROLES
│    • ∀res: RESOURCE | res ∈ RESOURCE
│        • res_grade . 1 = res ∧ res_grade . 2 = NIL
│          ∧ role_res_grade . 1 = role . 1 ∧ role_res_grade . 2 = role_res_grade . 2 ∪ {res_grade}
│ ∀res: RESOURCE | res ∈ {res_grade . 1}
│    • if ∃ne: NORMAL_ALLOC | ne . 1 = role_res_grade . 1 ∧ res ∈ ne . 2 • true
│      then res_grade . 2 = NORMAL
│      else if ∃oe: OVER_ALLOC | oe . 1 = role_res_grade . 1 ∧ res ∈ oe . 2 • true
│          then res_grade . 2 = OVER
│          else if ∃ue: UNDER_ALLOC | ue . 1 = role_res_grade . 1 ∧ res ∈ ue . 2 • true
│              then res_grade . 2 = UNDER
│              else res_grade . 2 = NIL
└────────────────────────────────────────
```

Figure 17: Specifications for grading resource for role.

Roles and resources are also generated with the given number as input. For the generated set of roles, a mapping is generated to map designation and charge to role with role assignment. For the given roles and resources, initially a policy is generated. Requests are generated by users and processed by the policy enforcement module, which logs each request in a log file. The log file is then used to create ResourceLog and RoleLog files, which are used as inputs for

```
ROLE_RESOURCE_CLUSTER == ROLES × 𝔽 RESOURCE
┌─ROLE_RESOURCE_CLUSTER_SET──────────────────
│ role_resource_cluster: ℙ ROLE_RESOURCE_CLUSTER
└────────────────────────────────────────────

┌─CREATE_ROLE_RES_CLUSTER────────────────────
│ ΞMONTH_LOG_SET
│ ΔROLE_RESOURCE_CLUSTER_SET
│ role_res_cluster: ROLE_RESOURCE_CLUSTER
│ stat: ℙ REQ_RES_STAT
│ res: ℙ RESOURCE
├──────────────────
│ ∀role: ROLE; month_log: MONTH_LOG | role . 1 = month_log . 2
│    • stat = month_log . 5 ∧ role_res_cluster . 1 = role . 1
│ ∀se: REQ_RES_STAT | se ∈ stat • res = res ∪ {se . 1}
│ role_resource_cluster' = role_resource_cluster ∪ {role_res_cluster}
└────────────────────────────────────────────
```

Figure 18: Specifications to create clusters.

```
ROLE_RES_COUNT == ROLES × ℤ
PROB == ℕ
ROLE_RES_PROB == ROLES × PROB
RES_ROLE_PROB_SET == RESOURCE × 𝔽 ROLE_RES_PROB
┌─RES_ROLE_PROB_SET_SCH──────────────────────
│ RES_ROLE_PROB_SET: ℙ RES_ROLE_PROB_SET
└────────────────────────────────────────────

RES_ROLE_COUNT == RESOURCE × ℤ
RES_ROLE_PROB == RESOURCE × PROB
ROLE_RES_PROB_SET == ROLES × 𝔽 RES_ROLE_PROB
┌─ROLE_RES_PROB_SET_SCH──────────────────────
│ ROLE_RES_PROB_SET: ℙ ROLE_RES_PROB_SET
└────────────────────────────────────────────

RES_WEIGHT == RESOURCE × ℤ
ROLE_RES_WEIGHT == ROLES × 𝔽 RES_WEIGHT
┌─ROLE_RES_WEIGHT_SET────────────────────────
│ ROLE_RES_WEIGHT: ℙ ROLE_RES_WEIGHT
└────────────────────────────────────────────
```

Figure 19: Specifications of types for weight calculation.

the proposed recommender algorithms. The recommendations are fed to the policy management module (PEM), which adapts the policies based on the recommendations. In the simulation, the same set of users, roles, and resources is used with different sets of requests to evaluate the impact of the recommendations on the system's performance. The next subsection presents the simulation outcomes, while the other subsection validates the simulation model.

A simulation study has been conducted to showcase the effectiveness of the proposed formal specifications, and various simulation parameters have been varied to observe their impact on the performance of the suggested mechanism. The simulation parameters are presented in Table 3.

Conventional RBAC is static in nature. An enterprise cloud is intended for an enterprise. The goal of an enterprise is to have the optimal availability of its resources to their users for maximal utilization of its resources with minimal availability. However, the subjective approach used in defining access control policies in traditional RBAC may have an impact on resource utilization. This effect is illustrated in Figure 31, where resources allowed for each role are categorized as normal, over, or under. A resource is considered normal if it appears in the user's request, over if it is never requested by the role's user, and under if it is in the requests but not available to users according to the defined policy. The figure displays the resource usage for one simulation scenario, showing that some resources are available for roles but are not requested, while others may be required by requests where they are not available.

So, the need for the present study arises. The present study suggests recommendations for revising the policies by extracting the role profile from the requests made by the users under each role. The role profile entails several aspects related to resource management within a specific role. The role profile identifies and lists the resources that are accessible and available to users assigned to that particular role. It also includes a record of the resources that users belonging to that role commonly request or require for their tasks or responsibilities. The role profile specifies the

```
__FIND_RES_ROLE_PROB______________________________
ΞMONTH_LOG_SET
ΞREQUEST_PROFILE
ΔRES_ROLE_PROB_SET_SCH
role_res_count: ROLE_RES_COUNT
role_res_count_set: 𝔽 ROLE_RES_COUNT
role_res_prob: ROLE_RES_PROB
role_res_prob_set: 𝔽 ROLE_RES_PROB
res_role_prob_set: RES_ROLE_PROB_SET
res_req: RESOURCE_REQUEST
cres: RESOURCE
total_count: ℤ
__________________
∀res: RESOURCE | res ∈ RESOURCE
  • ∀allroles: ROLES | allroles ∈ ROLES
      • ∀req: REQUEST | req ∈ REQUEST
          • if ∃role: ROLES | req . 3 = role ∧ role = allroles • true
            then if ∃res_req: RESOURCE_REQUEST | res_req ∈ req . 4 ∧ res_req . 1 = res • true
                then role_res_count . 1 = req . 3 ∧ cres = res
                    ∧ role_res_count . 2 = role_res_count . 2 + 1
                    ⇒ role_res_count_set = role_res_count_set ∪ {role_res_count}
                else role_res_count_set = role_res_count_set
            else role_res_count_set = role_res_count_set
∀rrc: ROLE_RES_COUNT | rrc ∈ role_res_count_set • total_count = total_count + rrc . 2
∀rrce: ROLE_RES_COUNT | rrce ∈ role_res_count_set
  • role_res_prob . 1 = rrce . 1 ∧ role_res_prob . 2 = rrce . 2
    ⇒ role_res_prob_set = role_res_prob_set ∪ {role_res_prob}
res_role_prob_set . 1 = cres ∧ res_role_prob_set . 2 = role_res_prob_set
RES_ROLE_PROB_SET′ = RES_ROLE_PROB_SET ∪ {res_role_prob_set}
__________________________________________________

__FIND_ROLE_RES_PROB______________________________
ΞMONTH_LOG_SET
ΞREQUEST_PROFILE
ΔROLE_RES_PROB_SET_SCH
res_role_count: RES_ROLE_COUNT
res_role_count_set: 𝔽 RES_ROLE_COUNT
res_role_prob: RES_ROLE_PROB
res_role_prob_set: 𝔽 RES_ROLE_PROB
role_res_prob_set: ROLE_RES_PROB_SET
res_req: RESOURCE_REQUEST
crole: ROLES
total_count: ℤ
__________________
∀role: ROLES | role ∈ ROLES
  • ∀res: RESOURCE | res ∈ RESOURCE
      • ∀req: REQUEST | req ∈ REQUEST
          • if ∃res_req: RESOURCE_REQUEST
                | res_req ∈ req . 4 ∧ res_req . 1 = res • true
            then res_role_count . 1 = res ∧ crole = role
                ∧ res_role_count . 2 = res_role_count . 2 + 1
                ⇒ res_role_count_set = res_role_count_set ∪ {res_role_count}
            else res_role_count_set = res_role_count_set
∀rrc: RES_ROLE_COUNT | rrc ∈ res_role_count_set • total_count = total_count + rrc . 2
∀rrce: RES_ROLE_COUNT | rrce ∈ res_role_count_set
  • res_role_prob . 1 = rrce . 1 ∧ res_role_prob . 2 = rrce . 2
    ⇒ res_role_prob_set = res_role_prob_set ∪ {res_role_prob}
role_res_prob_set . 1 = crole ∧ role_res_prob_set . 2 = res_role_prob_set
ROLE_RES_PROB_SET′ = ROLE_RES_PROB_SET ∪ {role_res_prob_set}
__________________________________________________

__CREATE_WEIGHTS__________________________________
ΞRES_ROLE_PROB_SET_SCH
ΞROLE_RES_PROB_SET_SCH
ΔROLE_RES_WEIGHT_SET
res_weight: RES_WEIGHT
res_weight_set: 𝔽 RES_WEIGHT
role_res_weight: ROLE_RES_WEIGHT
__________________
∀role: ROLES | role ∈ ROLES
  • ∀res: RESOURCE | res ∈ RESOURCE
      • res_weight . 1 = res ∧ res_weight . 2 = 0
        ⇒ res_weight_set = res_weight_set ∪ {res_weight} ∧ role_res_weight . 1 = role
          ∧ role_res_weight . 2 = res_weight_set
∀role: ROLES | role ∈ ROLES
  • ∀res: RESOURCE | res ∈ RESOURCE
      • ∃roleresprob: RES_ROLE_PROB_SET; resroleprob: ROLE_RES_PROB_SET
          | roleresprob ∈ RES_ROLE_PROB_SET
            ∧ resroleprob ∈ ROLE_RES_PROB_SET
            ∧ roleresprob . 1 = res ∧ resroleprob . 1 = role
          • ∃rorp: ROLE_RES_PROB; rerp: RES_ROLE_PROB
              | rorp . 1 = role ∧ rerp . 1 = res
              • ∃rrw: ROLE_RES_WEIGHT
                  | rrw ∈ {role_res_weight} ∧ rrw . 1 = role
                  • ∃rw: RES_WEIGHT | rw ∈ {res_weight} ∧ rw . 1 = res
                      • rw . 2 = rorp . 2 * rerp . 2
ROLE_RES_WEIGHT′ = ROLE_RES_WEIGHT ∪ {role_res_weight}
__________________________________________________
```

Figure 20: Specifications of creating weights for resource.

```
ROLE_RESOURCE_PTAGE == RESOURCE × ℤ
ROLE_RESOURCE_PTAGE_REQUEST == ROLES × 𝔽 ROLE_RESOURCE_PTAGE
┌─ROLE_RESOURCE_PTAGE_REQUEST_SET──────────────
│ ROLE_RESOURCE_PTAGE_REQUEST: ℙ ROLE_RESOURCE_PTAGE_REQUEST
└──────────────────────────────────────────────

┌─FIND_ROLE_RES_PTAGE──────────────────────────
│ ΞMONTH_LOG_SET
│ ΞREQUEST_PROFILE
│ ΔROLE_RESOURCE_PTAGE_REQUEST_SET
│ res_role_count: RES_ROLE_COUNT
│ res_role_count_set: 𝔽 RES_ROLE_COUNT
│ res_role_ptage: ROLE_RESOURCE_PTAGE
│ res_role_ptage_set: 𝔽 ROLE_RESOURCE_PTAGE
│ role_res_ptage_request: ROLE_RESOURCE_PTAGE_REQUEST
│ res_req: RESOURCE_REQUEST
│ crole: ROLES
│ total_count: ℤ
├──────────────────
│ ∀role: ROLES | role ∈ ROLES
│   • ∀res: RESOURCE | res ∈ RESOURCE
│       • ∀req: REQUEST | req ∈ REQUEST
│           • if ∃res_req: RESOURCE_REQUEST
│                   | res_req ∈ req . 4 ∧ res_req . 1 = res • true
│             then res_role_count . 1 = res
│                   ∧ crole = role
│                   ∧ res_role_count . 2 = res_role_count . 2 + 1
│                   ⇒ res_role_count_set
│                       = res_role_count_set ∪ {res_role_count}
│             else res_role_count_set = res_role_count_set
│ ∀rrc: RES_ROLE_COUNT | rrc ∈ res_role_count_set
│   • total_count = total_count + rrc . 2
│ ∀rrce: RES_ROLE_COUNT | rrce ∈ res_role_count_set
│   • res_role_ptage . 1 = rrce . 1 ∧ res_role_ptage . 2 = rrce . 2
│     ⇒ res_role_ptage_set = res_role_ptage_set ∪ {res_role_ptage}
│ role_res_ptage_request . 1 = crole
│ ∧ role_res_ptage_request . 2 = res_role_ptage_set
│ ROLE_RESOURCE_PTAGE_REQUEST′
│   = ROLE_RESOURCE_PTAGE_REQUEST ∪ {role_res_ptage_request}
└──────────────────────────────────────────────
```

Figure 21: Specification to find percentage.

```
┌─RECOMMENDATION_USING_GRADING─────────────────
│ ΔPOLICY_DATABASE
│ ΞROLE_RESOURCE_GRADING_SET
│ crole: ROLES
│ res_allow: RESOURCE_ALLOWED
│ res_allow_set: 𝔽 RESOURCE_ALLOWED
│ rev_policy: POLICY
├──────────────────
│ ∃role: ROLES; policy: POLICY | policy . 1 = role
│   • ∀ra: RESOURCE_ALLOWED | ra ∈ policy . 2
│       • ∀res: RESOURCE | res = ra . 1
│           • ∃rrg: ROLE_RES_GRADE | rrg ∈ ROLE_RES_GRADE ∧ rrg . 1 = role
│               • ∃rg: RES_GRADE | rg ∈ rrg . 2 ∧ rg . 1 = res
│                   • if rg . 2 = NORMAL
│                     then res_allow . 1 = rg . 1 ∧ res_allow . 2 = ra . 2
│                          ∧ crole = role ⇒ res_allow_set = res_allow_set ∪ {res_allow}
│                     else if rg . 2 = UNDER
│                          then res_allow . 1 = rg . 1 ∧ res_allow . 2 . INSTANCE = 3
│                               ∧ crole = role ⇒ res_allow_set = res_allow_set ∪ {res_allow}
│                          else if rg . 2 = OVER ∨ rg . 2 = NIL
│                               then res_allow = res_allow else res_allow = res_allow
│ rev_policy . 1 = crole ∧ rev_policy . 2 = res_allow_set ⇒ POLICY′ = POLICY ∪ {rev_policy}
└──────────────────────────────────────────────
```

Figure 22: Recommendations using resource grading.

```
RECOMMENDATION_USING_CLUSTER
$\Delta POLICY\_DATABASE$
$\Xi ROLE\_RESOURCE\_CLUSTER\_SET$
$crole: ROLES$
$res\_allow: RESOURCE\_ALLOWED$
$res\_allow\_set: \mathbb{F}\ RESOURCE\_ALLOWED$
$rev\_policy: POLICY$
────────────
$\exists role: ROLES; policy: POLICY \mid policy.1 = role$
    $\bullet\ \exists rrc: ROLE\_RESOURCE\_CLUSTER \mid rrc \in ROLE\_RESOURCE\_CLUSTER \wedge rrc.1 = role$
        $\bullet\ \forall res: RESOURCE \mid res \in rrc.2 \bullet res\_allow.1 = res \wedge res\_allow.2.INSTANCE = 3$
            $\wedge\ crole = role \Rightarrow res\_allow\_set = res\_allow\_set \cup \{res\_allow\}$
$rev\_policy.1 = crole \wedge rev\_policy.2 = res\_allow\_set \Rightarrow POLICY' = POLICY \cup \{rev\_policy\}$
```

Figure 23: Recommendations using clusters.

```
RECOMMENDATION_USING_WEIGHT
$\Delta POLICY\_DATABASE$
$\Xi ROLE\_RES\_WEIGHT\_SET$
$crole: ROLES$
$res\_allow: RESOURCE\_ALLOWED$
$res\_allow\_set: \mathbb{F}\ RESOURCE\_ALLOWED$
$rev\_policy: POLICY$
$wt\_thresh?: \mathbb{Z}$
────────────
$\exists role: ROLES; policy: POLICY \mid policy.1 = role$
    $\bullet\ \exists rrw: ROLE\_RES\_WEIGHT \mid rrw \in ROLE\_RES\_WEIGHT \wedge rrw.1 = role$
        $\bullet\ \forall rw: RES\_WEIGHT \mid rw \in rrw.2 \wedge rw.2 \geqslant wt\_thresh?$
            $\bullet\ res\_allow.1 = rw.1 \wedge res\_allow.2.INSTANCE = 3$
            $\wedge\ crole = role \Rightarrow res\_allow\_set = res\_allow\_set \cup \{res\_allow\}$
$rev\_policy.1 = crole \wedge rev\_policy.2 = res\_allow\_set$
$\Rightarrow POLICY' = POLICY \cup \{rev\_policy\}$
```

Figure 24: Recommendations using weight.

```
RECOMMENDATION_USING_PTAGE
$\Delta POLICY\_DATABASE$
$\Xi ROLE\_RESOURCE\_PTAGE\_REQUEST\_SET$
$crole: ROLES$
$res\_allow: RESOURCE\_ALLOWED$
$res\_allow\_set: \mathbb{F}\ RESOURCE\_ALLOWED$
$rev\_policy: POLICY$
$pt\_thresh?: \mathbb{Z}$
────────────
$\exists role: ROLES; policy: POLICY \mid policy.1 = role$
    $\bullet\ \exists rrpr: ROLE\_RESOURCE\_PTAGE\_REQUEST$
        $\mid rrpr \in ROLE\_RESOURCE\_PTAGE\_REQUEST \wedge rrpr.1 = role$
        $\bullet\ \forall rrp: ROLE\_RESOURCE\_PTAGE \mid rrp \in rrpr.2 \wedge rrp.2 \geqslant pt\_thresh?$
            $\bullet\ res\_allow.1 = rrp.1 \wedge res\_allow.2.INSTANCE = 3$
            $\wedge\ crole = role \Rightarrow res\_allow\_set = res\_allow\_set \cup \{res\_allow\}$
$rev\_policy.1 = crole \wedge rev\_policy.2 = res\_allow\_set$
$\Rightarrow POLICY' = POLICY \cup \{rev\_policy\}$
```

Figure 25: Recommendations using percentage.

current availability status of the resources for users assigned to that role. Conversely, the role profile also notes any resources that are unavailable or restricted for users in that role.

The application of the proposed recommendations yields a scenario shown in Figure 32 with maximum availability of resources to each role and that too with the same set of resources. A confusion matrix has been generated for each

```
STATE ::=  START
         | WAITING
         | ACCEPTED
         | DISCARDED
         | PROCESSING
         | COMPLETED
         | STOP
EVENT ::=  MAKE_REQUEST
         | REQUEST_SUBMISSION_ACM
         | REQUEST_FORWARD_CLOUD
         | REQUEST_COMPLETED_CLOUD
         | REJECT_REQUEST
         | GRADE_RESOURCE_AVAILABILITY
FSM  == STATE × EVENT ⇸ STATE
 | TRANSITION, NO_CHANGE: FSM
 |____________________________
 | NO_CHANGE = {  s: STATE; e: EVENT •  (s, e) ↦ s  }
 | TRANSITION
 |   = {(START, MAKE_REQUEST) ↦ WAITING,
 |      (WAITING, REQUEST_SUBMISSION_ACM) ↦ ACCEPTED,
 |      (WAITING, REQUEST_SUBMISSION_ACM) ↦ DISCARDED,
 |      (DISCARDED, REJECT_REQUEST) ↦ STOP,
 |      (ACCEPTED, REQUEST_FORWARD_CLOUD) ↦ PROCESSING,
 |      (PROCESSING, REQUEST_COMPLETED_CLOUD) ↦ COMPLETED,
 |      (COMPLETED, GRADE_RESOURCE_AVAILABILITY) ↦ STOP}
```

Figure 26: FSM of request.

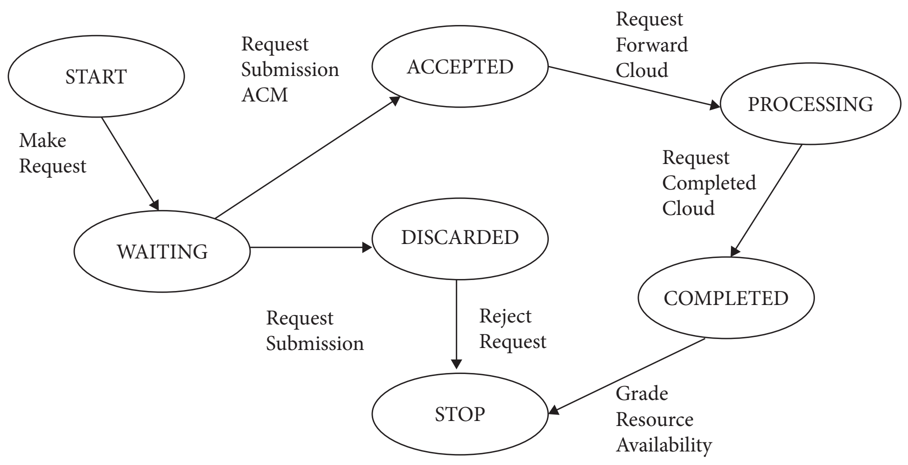


Figure 27: State transition diagram of request.

Table 1: State transition table for request.

	Start	Waiting	Accepted	Discarded	Processing	Completed	Stop
Start	—	Make_request	—	—	—	—	—
Waiting	—	—	Request submission ACM	Request submission ACM	—	—	—
Accepted	—	—	—	—	Request forward cloud	—	—
Discarded	—	—	—	—	—	—	Reject request
Processing	—	—	—	—	—	Request completed cloud	—
Completed	—	—	—	—	—	—	Grade resource availability
Stop	—	—	—	—	—	—	—

STATE ::= START | QUERY | UPDATE

EVENT ::= FindPolicy | AddPolicy | DeletePolicy | UpdatePolicy

FSM == STATE $\times$ EVENT $\twoheadrightarrow$ STATE

TRANSITION, NO_CHANGE: FSM

NO_CHANGE = { s: STATE; e: EVENT • (s, e) $\mapsto$ s }
TRANSITION
= {(START, FindPolicy) $\mapsto$ QUERY, (QUERY, AddPolicy) $\mapsto$ UPDATE,
(QUERY, DeletePolicy) $\mapsto$ UPDATE, (QUERY, UpdatePolicy) $\mapsto$ UPDATE,
(UPDATE, FindPolicy) $\mapsto$ QUERY}

Figure 28: Specification of FSM of policy.

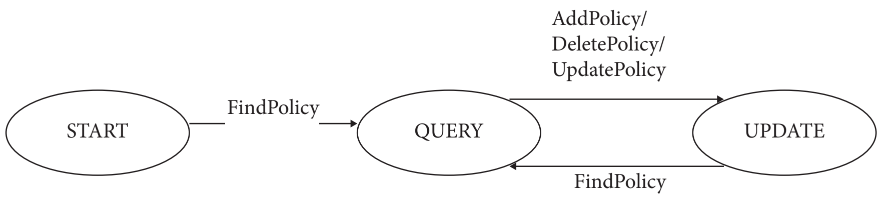


Figure 29: State transition diagram of policy database.

Table 2: State transition table for policy database.

	Start	Query	Update
Start	—	FindPolicy	—
Query	—	—	AddPolicy DeletePolicy UpdatePolicy
Update	—	FindPolicy	—

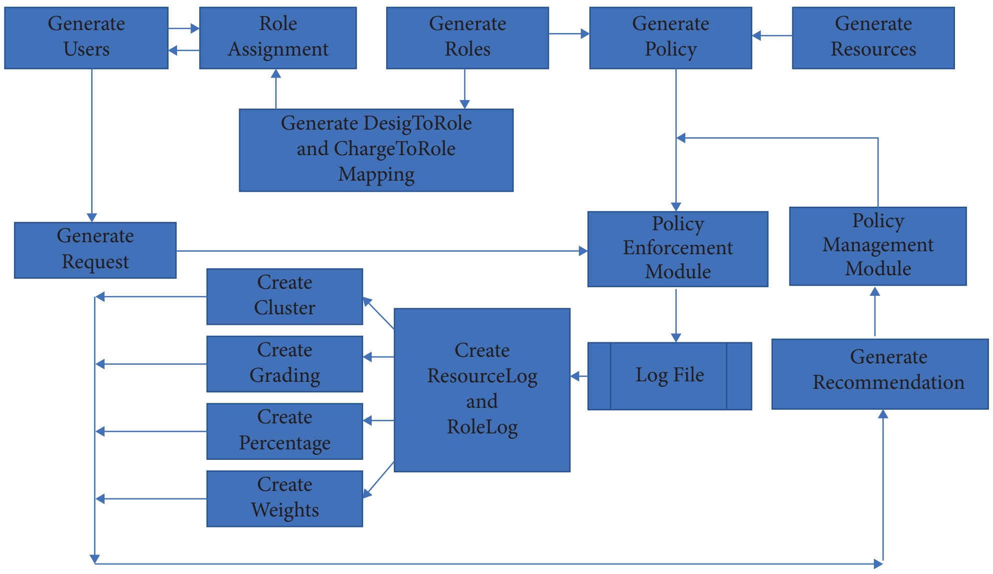


Figure 30: Simulation model.

Table 3: Simulation parameters.

Parameter	Value
No. of users	50, 200, 1000, 5000
No. of roles	20, 35, 50
No. of requests/day	10, 100, 100

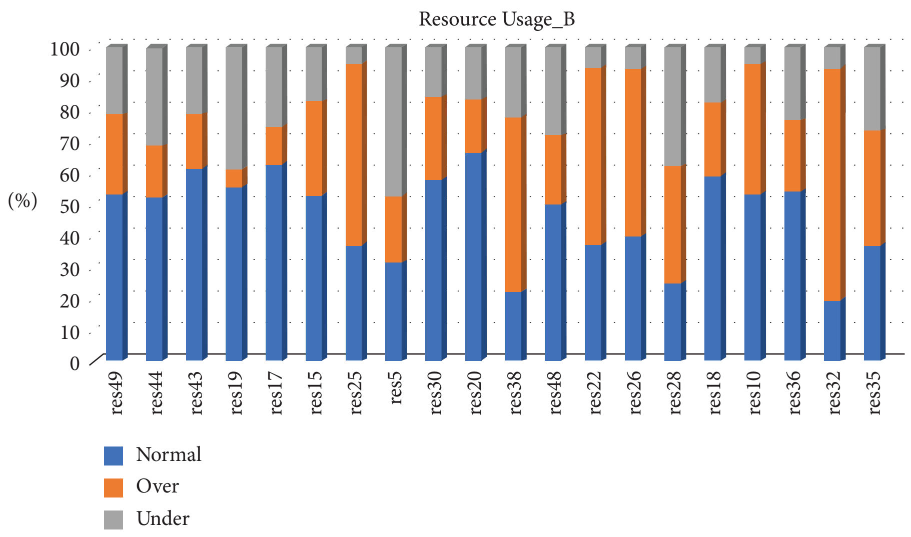


Figure 31: Resource usage before.

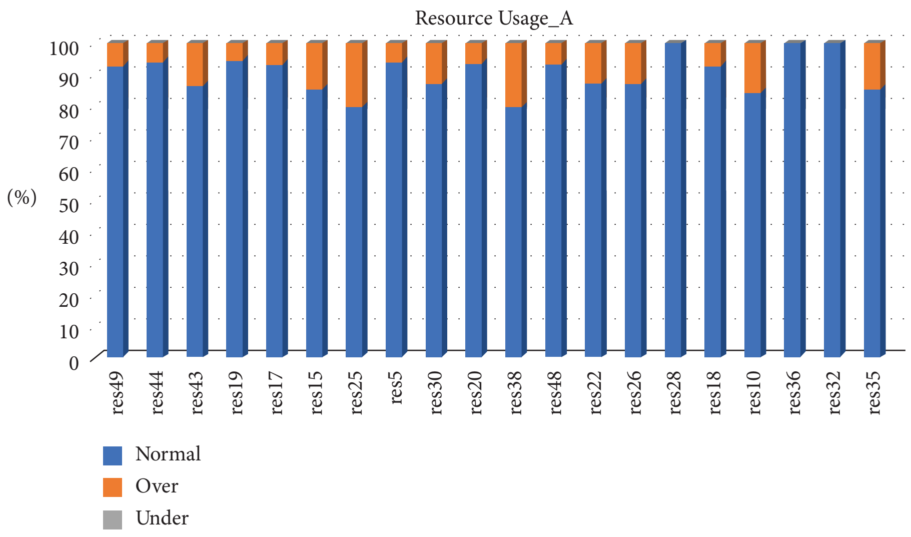


Figure 32: Resource usage after.

outcome of the simulation. The description of the entries of the matrix is as follows:

True positive: the number of resources required by the users of role is available in defined policy

True negative: the number of resources not required by the users of role is not available in defined policy

False positive: the number of resources not required by the users of role is available in defined policy

False negative: the number of resources required by the users of role is not available in defined policy

The performance of the suggested approach has been evaluated on the bases of the following performance metrics presented in Table 4.

The simulation study uses four approaches to validate the outcome of suggested study. These approaches are intended to build the role profile by mining the requests made by the users of that role and then propose recommendation on these bases. The brief idea of each approach is as follows:

Clustering approach: This approach is to make clusters of only those resources that are requested by the users for each role. A snapshot of sample output is shown in Figure 33.

Grading approach: The resources are graded as over, normal, and under for each role. The suggested recommendations are to keep resources with grade normal, remove resources with grade over, and add the resources with grade under. A sample output is shown in Figure 34.

Weight approach: The weights are assigned to each resource for every role. The weights are calculated on the basis of the probability of the role having a resource in their requests and the probability of resource request made by the role. Both of these probabilities are multiplied to get the weight. These weights are normalized in the range of 0 to 1. These weights are sorted in decreasing order for each role. The sorted list gives the recommendations of the resources in the order of their weights. A sample output is shown in Figure 35.

Percentage approach: The percentage is determined individually for each resource within each role by comparing the number of times the resource was requested to the total number of requests made by the users in that role. The calculated percentage is used for recommendation of the resources in the policy for every role. A sample output is shown in Figure 36.

Each approach is evaluated based on performance metrics. The first performance metric is the acceptance ratio of requests by the access control. Figure 37 shows the results before and after the adaptation of policies as per the recommendations. The figure demonstrates the results for every combination of simulation parameters, including the number of users, number of roles, and number of requests per day. It is observed from the figure that the acceptance ratio tremendously increases in all approaches. The performance of clustering and grading is comparable, while the outcome of the weight and percentage approaches is less. The reason for this is that in clustering and grading, all recommendations are adapted into policies, while restrictions are imposed on the weight and percentage approaches. The intention is to show the variant with and without restrictions.

To study the effect of the number of users on the results of the performance metrics, the proposed mechanism was simulated with 50, 200, 1000, and 5000 users, and the outputs are shown in Table 5. The values in the table reveal that the metric values increase with the increase in the number of users, but only to some extent. The main observation is that the mechanism consistently performs much better in all cases after adapting policies to the current behaviour of the users in terms of their requests.

The visual representation of the outputs of the above-mentioned explanation is shown in Figure 38. The bars representing the performance have large values for the metrics after adapting the policies with the suggested recommendations. Therefore, it can be concluded that the proposed mechanism performs well irrespective of the number of users.

The roles are very important as they determine the classes to which the resources are assigned. To investigate their effect on the performance metrics, a simulation study with 20, 35, and 50 possible roles was performed. The values for different numbers of roles are tabulated in Table 6. It is evident from the table that the proposed mechanism achieves better performance for all values of roles and in all approaches studied.

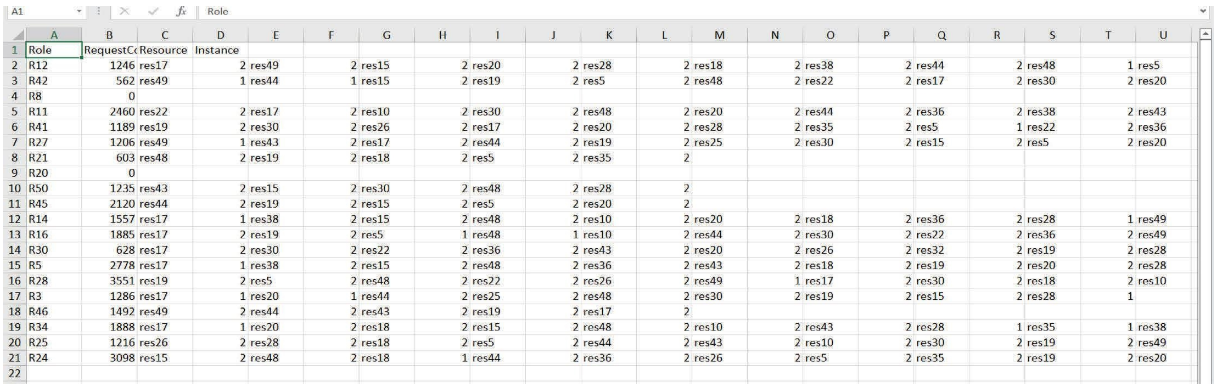
The outcomes in graphical form are shown in Figure 39. It can be observed from the representation that the values decrease in trend with the increase in roles. This is because with the increase in the number of roles, the number of classes also increases, and this increase has an effect on the value. However, even with the decreasing trend, the values are always better than before the adaptation of policies.

To investigate the effect of the number of requests on the outcomes of the proposed mechanism, a simulation study with 10, 100, and 1000 requests per day was conducted. The results are presented in Table 7, which shows that the proposed mechanism improves the metrics in all cases and in every approach. The increase in the number of requests leads to an increase in the values of the metrics, as expected. However, the improvement in performance is significant, indicating that the proposed mechanism is effective in handling large numbers of requests while maintaining high levels of security and resource availability.

The effects of the variation in the number of requests on the outcomes are visualized in Figure 40. It is observed that a smaller number of requests have a slightly lower improvement in performance, and it increases with an increase in the number of requests, but the performance improvement is very small after an increase in the number of requests. In the present simulation study, with 10 requests, the enhancement is less compared to 100 requests per day. Another increase to 1000 has no significant improvement compared to 100 requests.

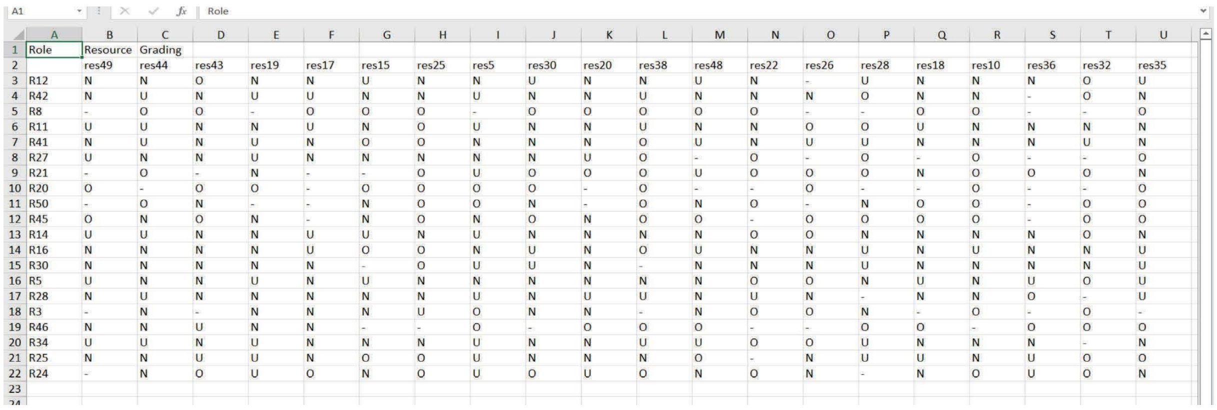
Table 4: Performance metrics.

Metric	Description
Acceptance ratio	The number of requests accepted and forwarded to cloud for further services
Accuracy	Ratio of correctly classified resources to the total no. of resources in each role
Precision	Ratio of correctly classified resources to the total classified resources in each role
Recall	Ratio of correctly classified resources to the actual required resources in each role
$F1$ measure	It is the harmonic mean of precision and recall. Beta represents how many times recall is more important than precision.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Role	RequestC	Resource	Instance																	
2	R12	1246	res17	2	res49	2	res15	2	res20	2	res28	2	res18	2	res38	2	res44	2	res48	1	res5
3	R42	562	res49	1	res44	1	res15	2	res19	2	res5	2	res48	2	res22	2	res17	2	res30	2	res20
4	R8	0																			
5	R11	2460	res22	2	res17	2	res10	2	res30	2	res48	2	res20	2	res44	2	res36	2	res38	2	res43
6	R41	1189	res19	2	res30	2	res26	2	res17	2	res20	2	res28	2	res35	2	res5	1	res22	2	res36
7	R27	1206	res49	1	res43	2	res17	2	res44	2	res19	2	res25	2	res30	2	res15	2	res5	2	res20
8	R21	603	res48	2	res19	2	res18	2	res5	2	res35	2									
9	R20	0																			
10	R50	1235	res43	2	res15	2	res30	2	res48	2	res28	2									
11	R45	2120	res44	2	res19	2	res15	2	res5	2	res20	2									
12	R14	1557	res17	1	res38	2	res15	2	res48	2	res10	2	res20	2	res18	2	res36	2	res28	1	res49
13	R16	1885	res17	2	res19	2	res5	1	res48	1	res10	2	res44	2	res30	2	res22	2	res36	2	res49
14	R30	628	res17	2	res30	2	res22	2	res36	2	res43	2	res20	2	res26	2	res32	2	res19	2	res28
15	R5	2778	res17	1	res38	2	res15	2	res48	2	res36	2	res43	2	res18	2	res19	2	res20	2	res28
16	R28	3551	res19	2	res5	2	res48	2	res22	2	res26	2	res49	1	res17	2	res30	2	res18	2	res10
17	R3	1286	res17	1	res20	1	res44	2	res25	2	res48	2	res30	2	res19	2	res15	2	res28	1	
18	R46	1492	res49	2	res44	2	res43	2	res19	2	res17	2									
19	R34	1888	res17	1	res20	2	res18	2	res15	2	res48	2	res10	2	res43	2	res28	1	res35	1	res38
20	R25	1216	res26	2	res28	2	res18	2	res5	2	res44	2	res43	2	res10	2	res30	2	res19	2	res49
21	R24	3098	res15	2	res48	2	res18	1	res44	2	res36	2	res26	2	res5	2	res35	2	res19	2	res20
22																					

Figure 33: Clustering approach.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Role	Resource	Grading																		
2		res49	res44	res43	res19	res17	res15	res25	res5	res30	res20	res38	res48	res22	res26	res28	res18	res10	res36	res32	res35
3	R12	N	N	O	N	N	U	N	N	U	N	N	U	N	-	U	N	N	N	O	U
4	R42	N	U	N	U	U	N	N	U	N	N	U	N	N	N	O	N	N	-	O	N
5	R8	-	O	O	-	O	O	O	-	O	O	O	O	O	-	-	O	O	-	-	O
6	R11	U	U	N	N	U	N	O	U	N	N	U	N	N	O	O	U	N	N	N	N
7	R41	N	U	N	U	N	O	O	N	N	N	O	U	N	U	U	N	N	N	U	N
8	R27	U	N	N	U	N	N	N	N	N	U	O	-	O	-	O	-	O	-	-	O
9	R21	-	O	-	N	-	-	O	U	O	O	O	U	O	O	O	N	O	O	O	N
10	R20	O	-	O	O	-	O	O	O	O	-	O	-	-	O	-	-	O	-	-	O
11	R50	-	O	N	-	-	N	O	O	N	-	O	N	O	-	N	O	O	-	O	O
12	R45	O	N	O	N	-	N	O	N	O	N	O	O	-	O	O	O	O	-	O	O
13	R14	U	U	N	N	U	U	N	U	N	N	N	N	O	O	N	N	N	N	O	N
14	R16	N	N	N	N	U	O	O	N	U	N	O	U	N	N	U	N	U	N	N	U
15	R30	N	N	N	N	N	-	O	U	U	N	-	N	N	N	U	N	N	N	N	U
16	R5	U	N	N	U	N	U	N	N	N	N	N	N	O	O	N	U	N	U	O	U
17	R28	N	U	N	N	N	N	N	U	N	U	U	N	U	N	-	N	N	O	-	U
18	R3	-	N	-	N	N	N	U	O	N	N	-	N	O	O	N	-	O	-	O	-
19	R46	N	N	U	N	N	-	-	O	-	O	O	O	-	-	O	O	-	O	O	O
20	R34	U	U	N	U	N	N	N	U	N	N	U	U	O	O	U	N	N	N	-	N
21	R25	N	N	U	U	N	O	O	U	N	N	N	O	-	N	U	U	N	U	O	O
22	R24	-	N	O	U	O	N	O	U	O	U	O	N	O	N	-	N	O	U	O	N
23																					

Figure 34: Grading approach.

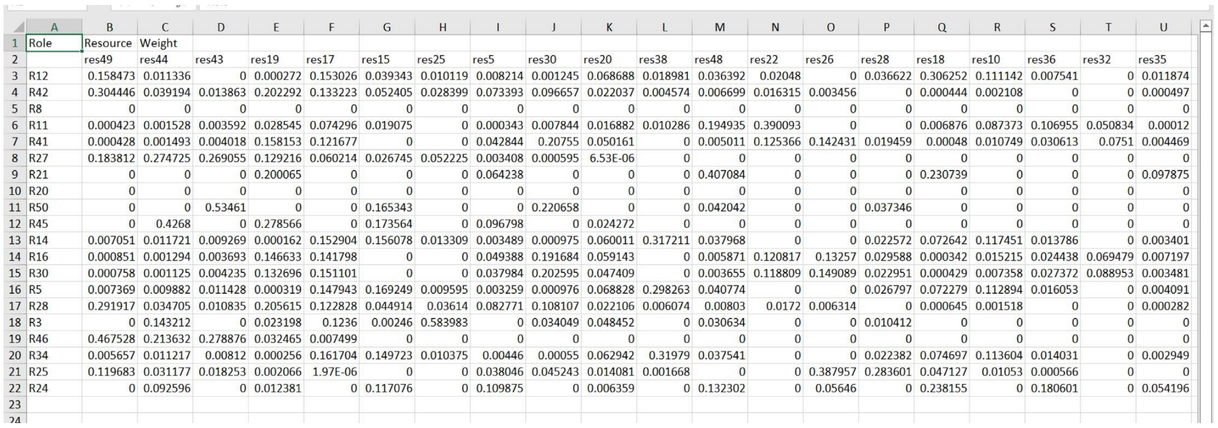
4.2. Verification and Validation of Simulation Model. Model validation aims to verify that the model is actually performing satisfactorily in its application area. In this case, the model was developed to demonstrate a recommender system for adapting access control policies to prevent over-allocation and under-allocation of resources. The outcomes of the trials and evaluations conducted during the model development phases were validated and verified to ensure that the model's performance meets the expected level of satisfaction. The following techniques are used for validation:

Degeneracy tests: to prove the degeneracy of the behaviour of model, it is tested with various combination of number of users, number of roles, and number of requests.

Event validity: acceptance or rejection of requests by policy enforcement module and revision of the policies are some of the events that happen in the model, and the occurrence of such events is similar to the real system.

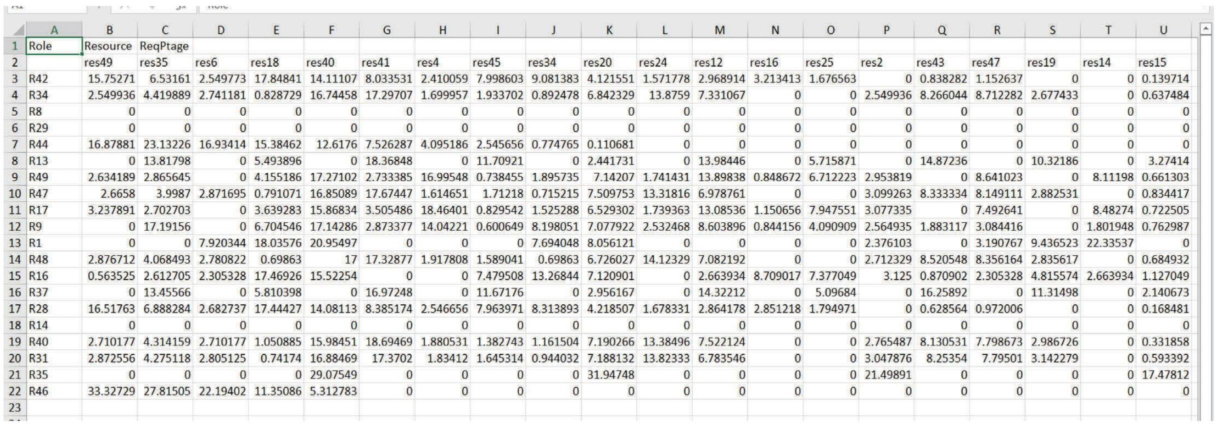
Extreme condition tests: the conditions like no request generated by a role, request of more than permissible instances of an allowed resource, and having multiple roles are considered and the handling mechanism is implemented.

Internal validity and parameter variability: a number of simulations for the same set of users, roles, and resources but different set of requests that are of the same number as in other simulation show the consistent behaviour.



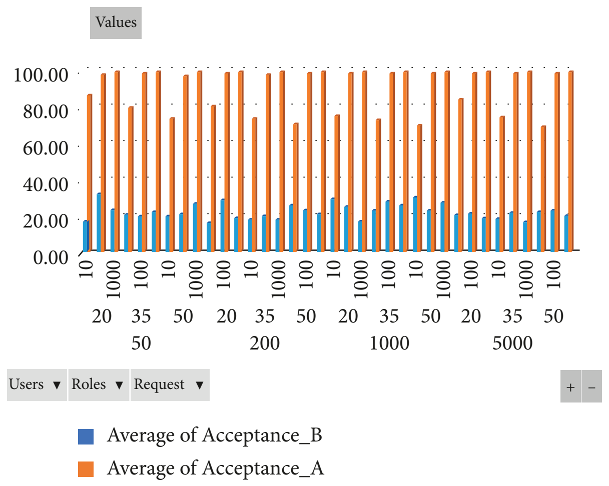
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Role	Resource	Weight																		
2		res49	res44	res43	res19	res17	res15	res25	res5	res30	res20	res38	res48	res22	res26	res28	res18	res10	res36	res32	res35
3	R12	0.158473	0.011336	0	0.000272	0.153026	0.039343	0.010119	0.008214	0.001245	0.068688	0.018981	0.036392	0.02048	0	0.036622	0.306252	0.111142	0.007541	0	0.011874
4	R42	0.304446	0.039194	0.013863	0.202292	0.133223	0.052405	0.028399	0.073393	0.096657	0.022037	0.004574	0.006699	0.016315	0.003456	0	0.000444	0.002108	0	0	0.000497
5	R8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	R11	0.000423	0.001528	0.003592	0.028545	0.074296	0.019075	0	0.000343	0.007844	0.016882	0.010286	0.194935	0.390093	0	0	0.006876	0.087373	0.106955	0.050834	0.00012
7	R41	0.000428	0.001493	0.004018	0.158153	0.121677	0	0	0.042844	0.20755	0.050161	0	0.005011	0.125366	0.142431	0.019459	0.00048	0.010749	0.030613	0.0751	0.004469
8	R27	0.183812	0.274725	0.269055	0.129216	0.060214	0.026745	0.052225	0.003408	0.000595	6.53E-06	0	0	0	0	0	0	0	0	0	0
9	R21	0	0	0	0.200065	0	0	0	0.064238	0	0	0	0.407084	0	0	0	0.230739	0	0	0	0.097875
10	R20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	R50	0	0	0.53461	0	0	0.165343	0	0	0.220658	0	0	0.042042	0	0	0.037346	0	0	0	0	0
12	R45	0	0.4268	0	0.278566	0	0.173564	0	0.096798	0	0.024272	0	0	0	0	0	0	0	0	0	0
13	R14	0.007051	0.011721	0.009269	0.000162	0.152904	0.156078	0.013309	0.003489	0.000975	0.060011	0.317211	0.037968	0	0	0.022572	0.072642	0.117451	0.013786	0	0.003401
14	R16	0.000851	0.001294	0.003693	0.146633	0.141798	0	0	0.049388	0.191684	0.059143	0	0.005871	0.120817	0.13257	0.029588	0.000342	0.015215	0.024438	0.069479	0.007197
15	R30	0.000758	0.001125	0.004235	0.132696	0.151101	0	0	0.037984	0.202595	0.047409	0	0.003655	0.118809	0.149089	0.022951	0.000429	0.007358	0.027372	0.088953	0.003481
16	R5	0.007369	0.009882	0.011428	0.000319	0.147943	0.169249	0.009595	0.003259	0.000976	0.068828	0.298263	0.040774	0	0	0.026797	0.072279	0.112894	0.016053	0	0.004091
17	R28	0.291917	0.034705	0.010835	0.205615	0.122828	0.044914	0.03614	0.082771	0.108107	0.022106	0.006074	0.00803	0.0172	0.006314	0	0.000645	0.001518	0	0	0.000282
18	R3	0	0.143212	0	0.023198	0.1236	0.00246	0.583983	0	0.034049	0.048452	0	0.030634	0	0	0.010412	0	0	0	0	0
19	R46	0.467528	0.213632	0.278876	0.032465	0.007499	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	R34	0.005657	0.011217	0.00812	0.000256	0.161704	0.149723	0.010375	0.00446	0.00055	0.062942	0.31979	0.037541	0	0	0.022382	0.074697	0.113604	0.014031	0	0.002949
21	R25	0.119683	0.031177	0.018253	0.002066	1.97E-06	0	0	0.038046	0.045243	0.014081	0.001668	0	0	0.387957	0.283601	0.047127	0.01053	0.000566	0	0
22	R24	0	0.092596	0	0.012381	0	0.117076	0	0.109875	0	0.006359	0	0.132302	0	0.05646	0	0.238155	0	0.180601	0	0.054196
23																					

Figure 35: Weight approach.

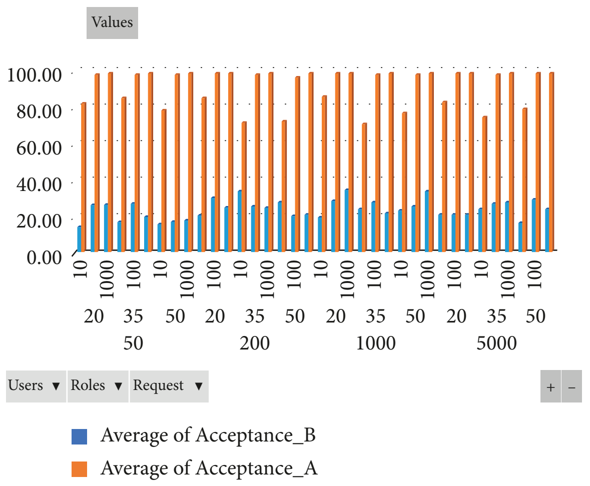


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Role	Resource	ReqPtage																		
2		res49	res35	res6	res18	res40	res41	res4	res45	res34	res20	res24	res12	res16	res25	res2	res43	res47	res19	res14	res15
3	R42	15.75271	6.53161	2.549773	17.84841	14.11107	8.033531	2.410059	7.998603	9.081383	4.121551	1.571778	2.968914	3.213413	1.676563	0	0.838282	1.152637	0	0	0.139714
4	R34	2.549936	4.419889	2.741181	0.828729	16.74458	17.29707	1.699957	1.933702	0.892478	6.842329	13.8759	7.331067	0	0	2.549936	8.266044	8.712282	2.677433	0	0.637484
5	R8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	R29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	R44	16.87881	23.13226	16.93414	15.38462	12.6176	7.526287	4.095186	2.545656	0.774765	0.110681	0	0	0	0	0	0	0	0	0	0
8	R13	0	13.81798	0	5.493896	0	18.36848	0	11.70921	0	2.441731	0	13.98446	0	5.715871	0	14.87236	0	10.32186	0	3.27414
9	R49	2.634189	2.865645	0	4.155186	17.27102	2.733385	16.99548	0.738455	1.895735	7.14207	1.741431	13.89838	0.848672	6.712223	2.953819	0	8.641023	0	8.11198	0.661303
10	R47	2.6658	3.9987	2.871695	0.791071	16.85089	17.67447	1.614651	1.71218	0.715215	7.509753	13.31816	6.978761	0	0	3.099263	8.333334	8.149111	2.882531	0	0.834417
11	R17	3.237891	2.702703	0	3.639283	15.86834	3.505486	18.46401	0.829542	1.525288	6.529302	1.739363	13.08536	1.150656	7.947551	3.077335	0	7.492641	0	8.48274	0.722505
12	R9	0	17.19156	0	6.704546	17.14286	2.873377	14.04221	0.600649	8.198051	7.077922	2.532468	8.603896	0.844156	4.090909	2.564935	1.883117	3.084416	0	1.801948	0.762987
13	R1	0	0	7.920344	18.03576	20.95497	0	0	0	7.694048	8.056121	0	0	0	0	2.376103	0	3.190767	9.436523	22.33537	0
14	R48	2.876712	4.068493	2.780822	0.69863	17	17.32877	1.917808	1.589041	0.69863	6.726027	14.12329	7.082192	0	0	2.712329	8.520548	8.356164	2.835617	0	0.684932
15	R16	0.563525	2.612705	2.305328	17.46926	15.52254	0	0	7.479508	13.26844	7.120901	0	2.663934	8.709017	7.377049	3.125	0.870902	2.305328	4.815574	2.663934	1.127049
16	R37	0	13.45566	0	5.810398	0	16.97248	0	11.67176	0	2.956167	0	14.32212	0	5.09684	0	16.25892	0	11.31498	0	2.140673
17	R28	16.51763	6.888284	2.682737	17.44427	14.08113	8.385174	2.546656	7.963971	8.313893	4.218507	1.678331	2.864178	2.851218	1.794971	0	0.628564	0.972006	0	0	0.168481
18	R14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	R40	2.710177	4.314159	2.710177	1.050885	15.98451	18.69469	1.880531	1.382743	1.161504	7.190266	13.38496	7.522124	0	0	2.765487	8.130531	7.798673	2.986726	0	0.331858
20	R31	2.872556	4.275118	2.805125	0.74174	16.88469	17.3702	1.83412	1.645314	0.944032	7.188132	13.82333	6.783546	0	0	3.047876	8.25354	7.79501	3.142279	0	0.593392
21	R35	0	0	0	0	29.07549	0	0	0	0	31.94748	0	0	0	0	21.49891	0	0	0	0	17.47812
22	R46	33.32729	27.81505	22.19402	11.35086	5.312783	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23																					

Figure 36: Percentage approach.

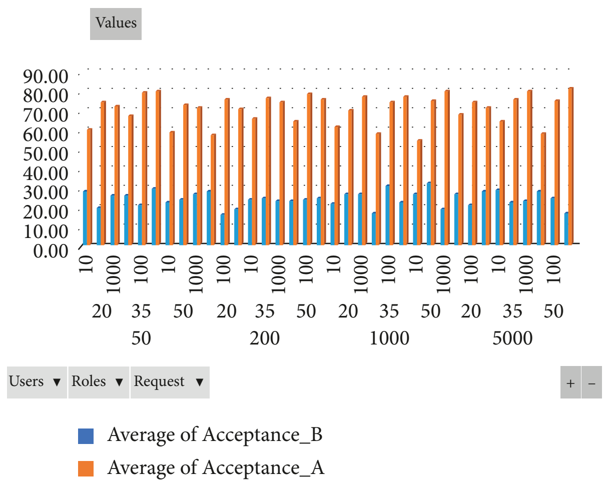


(a)

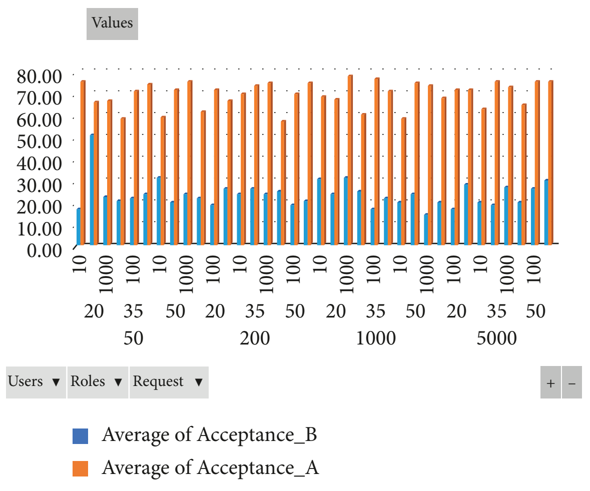


(b)

Figure 37: Continued.



(c)



(d)

Figure 37: Acceptance ratio before and after. (a) Clustering approach. (b) Grading approach. (c) Weight approach. (d) Percentage approach.

Table 5: Users vs. performance metrics.

Approach	Users	Accur_*B*	Accur_*A*	Recall_*B*	Recall_*A*	Precision_*B*	Precision_*A*	*F*1 score_*B*	*F*1 score_*A*
Cluster	50	0.46	0.75	0.47	0.64	0.38	0.66	0.39	0.65
	200	0.48	0.82	0.55	0.75	0.46	0.80	0.47	0.77
	1000	0.49	0.84	0.55	0.77	0.47	0.81	0.48	0.78
	5000	0.46	0.77	0.49	0.66	0.38	0.70	0.40	0.68
Grade	50	0.45	0.74	0.43	0.60	0.35	0.64	0.36	0.62
	200	0.49	0.83	0.57	0.76	0.46	0.79	0.48	0.77
	1000	0.49	0.84	0.56	0.77	0.47	0.80	0.48	0.78
	5000	0.45	0.74	0.45	0.61	0.37	0.64	0.38	0.62
Weight	50	0.46	0.58	0.45	0.42	0.39	0.64	0.40	0.50
	200	0.48	0.66	0.54	0.53	0.47	0.79	0.47	0.62
	1000	0.50	0.67	0.57	0.55	0.48	0.81	0.50	0.64
	5000	0.46	0.61	0.48	0.47	0.39	0.68	0.41	0.54
Percentage	50	0.45	0.73	0.45	0.28	0.38	0.65	0.39	0.36
	200	0.48	0.68	0.54	0.35	0.46	0.81	0.47	0.45
	1000	0.51	0.66	0.58	0.33	0.49	0.81	0.51	0.44
	5000	0.49	0.69	0.54	0.35	0.45	0.80	0.47	0.45

Operational graphics: the dynamism in the values of performance indicators is visualized to confirm the correctness of the mechanism.

Predictive validation: The mechanism is to suggest the recommendations by forecasting the behaviours of users in every role. The improvement in acceptance and other performance metrics validates the predictive ability of the model.

Traces: The outcomes of various kinds of internal processing are recorded and used for further processing in the generation of recommendations. The acceptance ratio and confusion matrix entries are some of the traces used in the proposed system.

So, on the bases of these techniques, the simulation model is operationally validated.

To elaborate further, the mechanism presented in the paper involves regularly analyzing the log entries that encompass the status of user requests for computing resources in the enterprise cloud. The analysis is done to identify over- and under-utilization of resources as per the policies defined by the enterprise for each user role. Based on the analysis, the access control policies are redefined to optimize the utilization of resources and improve the acceptance rate of requests. By optimizing the utilization of computing resources, the proposed mechanism helps to reduce over-allocation and under-allocation of resources to different roles within the enterprise, which can result in improved speed, performance, and utilization of computing resources. Additionally, the mechanism can provide insights into which resources are no longer needed or not frequently used, allowing for more efficient management of resources in the

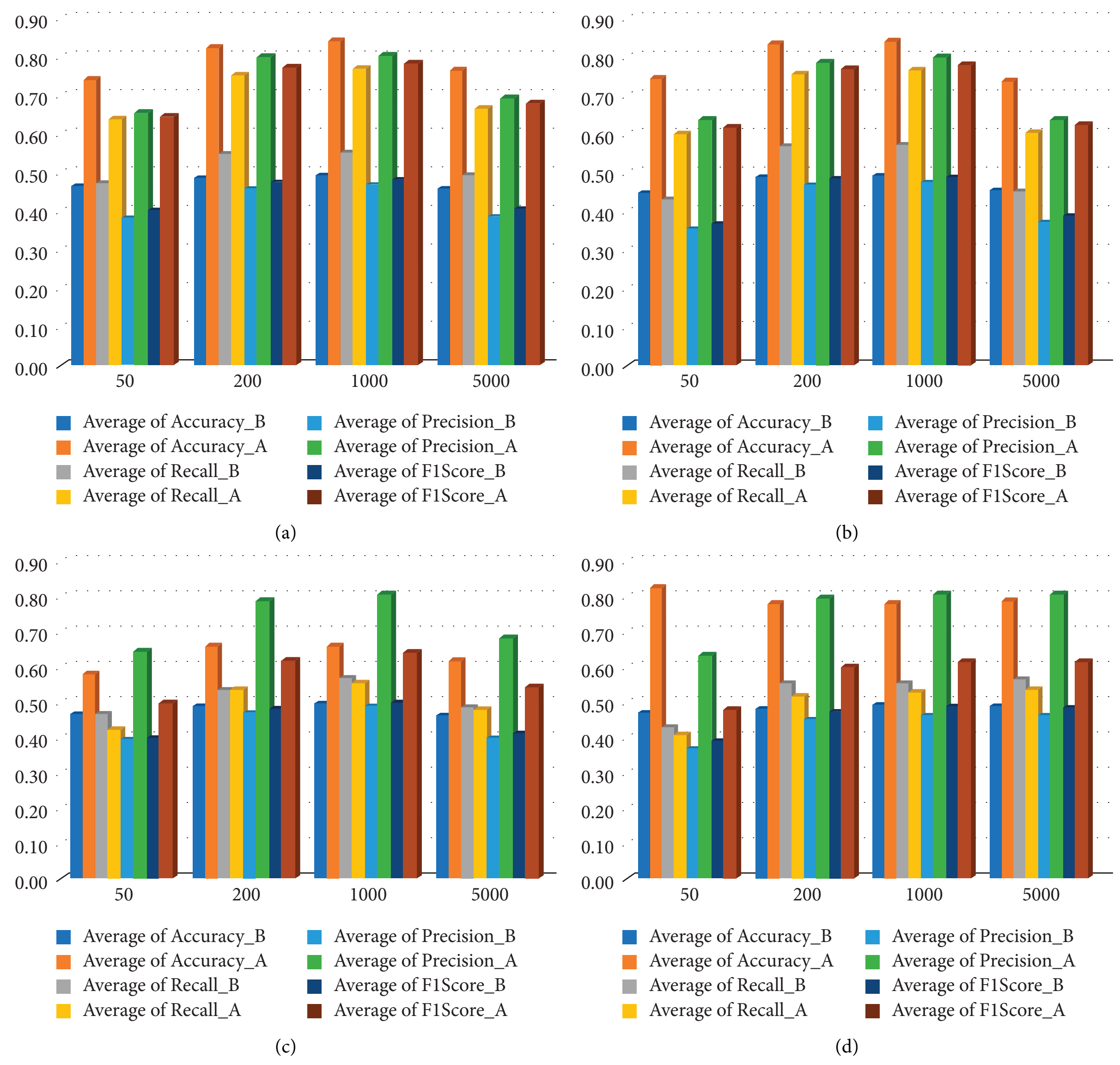
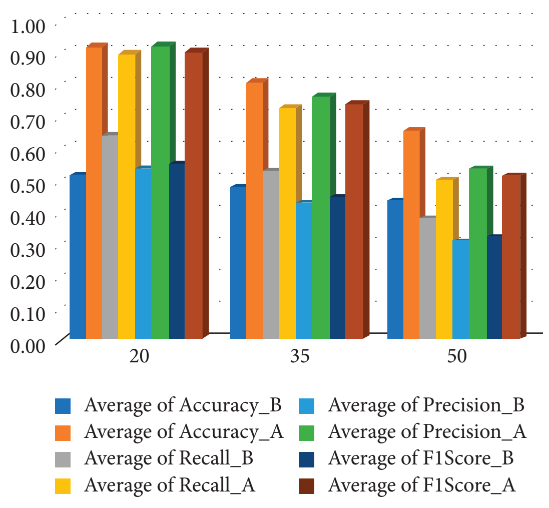


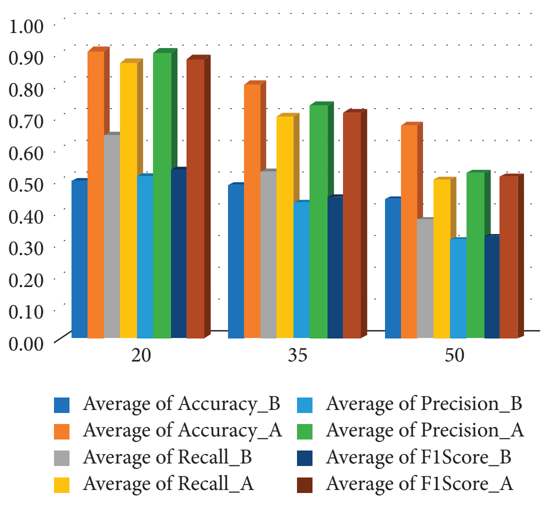
Figure 38: Users vs. performance metrics. (a) Clustering. (b) Grading. (c) Weight. (d) Percentage.

Table 6: Roles vs. performance metrics.

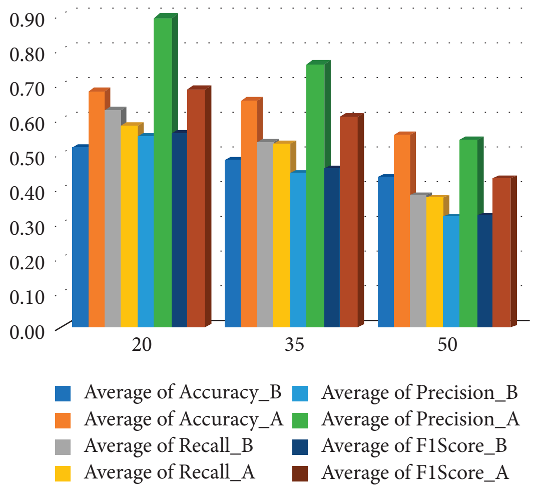
Approach	Roles	Accur_*B*	Accur_*A*	Recall_*B*	Recall_*A*	Precision_*B*	Precision_*A*	*F*1 score_*B*	*F*1 score_*A*
Cluster	20	0.51	0.92	0.64	0.90	0.53	0.92	0.55	0.91
	35	0.48	0.81	0.53	0.72	0.43	0.76	0.45	0.74
	50	0.43	0.66	0.38	0.50	0.30	0.53	0.32	0.51
Grade	20	0.49	0.91	0.62	0.87	0.51	0.90	0.53	0.88
	35	0.48	0.80	0.52	0.70	0.42	0.73	0.44	0.71
	50	0.44	0.67	0.37	0.49	0.30	0.52	0.32	0.50
Weight	20	0.52	0.68	0.63	0.58	0.55	0.89	0.56	0.69
	35	0.48	0.65	0.53	0.53	0.44	0.76	0.46	0.61
	50	0.43	0.56	0.38	0.37	0.31	0.54	0.32	0.43
Percentage	20	0.52	0.74	0.65	0.60	0.55	0.94	0.57	0.71
	35	0.48	0.79	0.54	0.52	0.44	0.79	0.46	0.61
	50	0.43	0.85	0.38	0.36	0.31	0.55	0.32	0.42



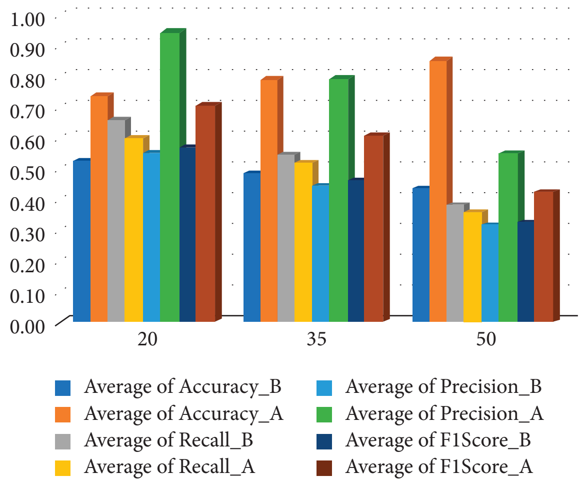
(a)



(b)



(c)



(d)

Figure 39: Roles vs. performance metrics. (a) Clustering. (b) Grading. (c) Weight. (d) Percentage.

Table 7: Requests/day vs. performance metrics.

Approach	Req/day	Accur_*B*	Accur_*A*	Recall_*B*	Recall_*A*	Precision_*B*	Precision_*A*	*F*1 score_*B*	*F*1 score_*A*
Cluster	10	0.45	0.74	0.52	0.64	0.35	0.73	0.40	0.67
	100	0.49	0.83	0.53	0.76	0.46	0.77	0.47	0.76
	1000	0.48	0.81	0.49	0.72	0.45	0.73	0.44	0.72
Grade	10	0.44	0.74	0.53	0.63	0.34	0.71	0.40	0.66
	100	0.48	0.82	0.50	0.73	0.46	0.74	0.45	0.73
	1000	0.48	0.81	0.48	0.70	0.44	0.70	0.43	0.70
Weight	10	0.45	0.66	0.54	0.52	0.37	0.74	0.42	0.60
	100	0.49	0.63	0.51	0.50	0.48	0.75	0.47	0.58
	1000	0.48	0.60	0.49	0.46	0.45	0.70	0.44	0.54
Percentage	10	0.44	0.84	0.52	0.53	0.34	0.74	0.39	0.60
	100	0.49	0.77	0.52	0.47	0.47	0.77	0.47	0.56
	1000	0.50	0.77	0.54	0.47	0.49	0.77	0.49	0.57

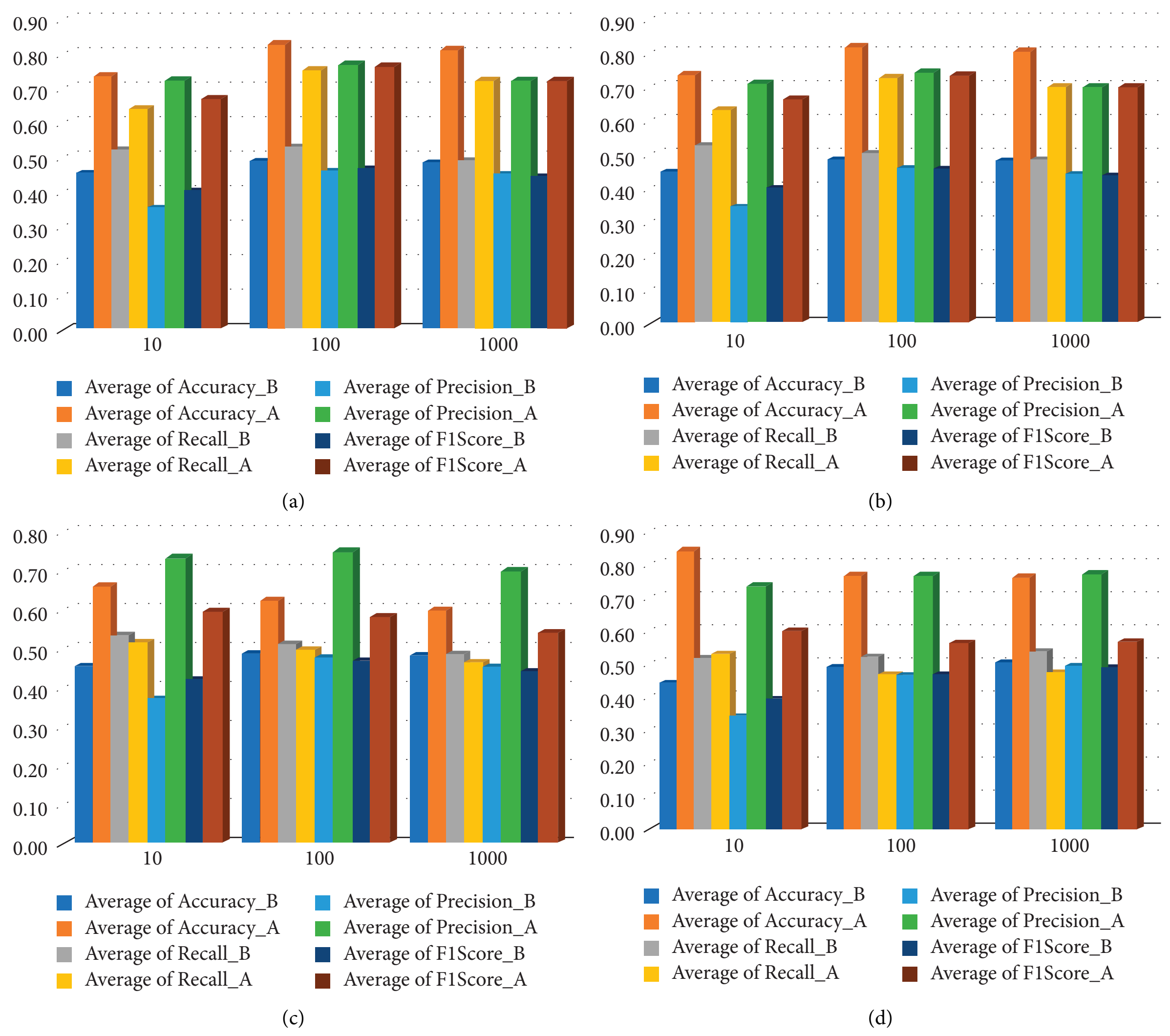


Figure 40: Requests/day vs. performance metrics. (a) Clustering. (b) Grading. (c) Weight. (d) Percentage.

enterprise cloud. Overall, the proposed mechanism is valuable in dynamic environments where the demand for computing resources can fluctuate rapidly. By regularly analyzing and redefining access control policies, the mechanism enables the enterprise to adapt to changing scenarios and ensure that computing resources are allocated efficiently and effectively to different user roles. This can ultimately result in improved operational capabilities and lower costs for the enterprise.

5. Conclusion

In this paper, the authors propose a mechanism designed to improve the utilization of computing resources in an enterprise by regularly redefining access control policies. The mechanism is particularly valuable in dynamic environments where the system must adapt to changing scenarios. The authors explain that in traditional enterprise systems, computing resources are physically and dedicatedly allocated to specific users or roles. However, in an enterprise cloud computing environment, resources are not allocated in this way. Instead, resources are provided logically to users based on their needs, as specified in their requests, but are subject to access control policies defined by the enterprise. These policies determine which users have access to which resources based on factors such as the role of the user, the type of data or application being accessed, and the kind of device being used.

The proposed mechanism is designed to analyze log entries that encompass the status of user requests for each role in order to identify over- and under-availability of resources as per the access control policies defined by the enterprise. This analysis is intended to lead to the redefinition of access control policies in order to increase overall resource utilization and optimal availability of resources to enterprise users. The mechanism is described in detail, including a state transition diagram and table that depict its behaviour. The authors explain that the mechanism enhances operational capabilities by reducing over-allocation and under-allocation of resources to roles. The reports generated by the mechanism can also aid in decision making about the need for additional instances of resources and identify resources that are no longer needed or not frequently used.

Overall, the proposed mechanism is intended to improve the utilization of computing resources in an enterprise by dynamically redefining access control policies based on user requests and role profiles. By increasing the availability of resources to users and roles that need them and reducing over-allocation and under-allocation, the mechanism is expected to lead to improved operational capabilities with less number of resources and increased overall resource utilization.

Data Availability

The results presented in this study are based on simulations. The underlying data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Samarati and S. C. de Vimercati, "Access control: policies, models, and mechanisms," *Foundations of Security Analysis and Design*, vol. 2171, pp. 137–196, 2001.

[2] R. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[3] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, "Assessment of access control systems," *NISTIR 7316*, 2006.

[4] V. Suhendra, "A survey on access control deployment," *Security Technology*, vol. 259, pp. 11–20, 2011.

[5] B. W. Lampson, "Protection," *ACM SIGOPS-Operating Systems Review*, vol. 8, no. 1, pp. 18–24, 1974.

[6] J. Rushby, "The Bell and La padula security model," Draft report, Computer Science Laboratory, Karamadai, TN, India, 1986.

[7] N. Meghanathan, "Review of access control models for cloud computing," *Computer Science & Information Technology (CS & IT)*, vol. 3, pp. 77–85, 2013.

[8] Y. A Younis, K. Kifayat, and M. Merabti, "An access control model for cloud computing," *Journal of Information Security and Applications*, vol. 19, no. 1, pp. 45–60, 2014.

[9] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," in *Proceedings of the 15th National Computer Security Conference*, pp. 554–563, Baltimore, MA, USA, August 1992.

[10] Y. Zhao, Y. Zhao, and H. Lu, "A flexible role- and resource-based access control model," in *Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, pp. 75–79, Guangzhou, China, August 2008.

[11] M. Yagüe, M.-M. M. Gallardo, and A. Maña, "Semantic access control model: a formal specification," *Computer Security–ESORICS*, vol. 3679, pp. 24–43, 2005.

[12] A. R. Khan, "Access control in cloud computing environment," *ARPN Journal of Engineering and Applied Sciences*, vol. 7, no. 5, pp. 613–615, 2012.

[13] E. Damiani, S. D. C. di Vimercati, and P. Samarati, "New paradigms for access control in open environments. Signal Processing and Information Technology," in *Proceedings of the Fifth IEEE International Symposium On*, pp. 540–545, Toronto, ON, Canada, August 2005.

[14] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, "A novel attribute-based access control scheme using blockchain for IoT," *IEEE Access*, vol. 7, pp. 38431–38441, 2019.

[15] A. Majumdar, S. Namasudra, and S. Nath, "Taxonomy and classification of access control models for cloud computing," in *Continued Rise of the Cloud, Computer Communications and Networks*, Z. Mahmood, Ed., pp 23–32, Springer, Berlin, Germany, 2014.

[16] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila, "A semantic context-aware access control framework for secure collaborations in pervasive computing environments," *Lecture Notes in Computer Science*, vol. 4273, pp. 473–486, 2006.

[17] A. S. M. Kayes, R. Kalaria, I. H. Sarker et al., "A survey of context-aware access control mechanisms for cloud and fog networks: taxonomy and open research issues," *Sensors*, vol. 20, no. 9, p. 2464, 2020.

[18] M. Gupta, Maanak, and S.. Ravi, "Towards activity-centric access control for smart collaborative ecosystems," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, pp. 155–164, Spain, June 2021.

[19] G. Cathey, J. Benson, M. Gupta, and R. Sandhu, "Edge centric secure data sharing with digital twins in smart ecosystems," in *Proceedings of the 2021 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pp. 70–79, Atlanta, GA, USA, December 2021.

[20] L. Praharaj, S. Ameer, M. Gupta, and R. Sandhu, "Attributes aware relationship-based access control for smart IoT systems," in *Proceedigs of the 2022 IEEE 8th International Conference on Collaboration and Internet Computing (CIC)*, pp. 72–81, Atlanta, GA, USA, August 2022.

[21] M. Gupta, R. Sandhu, T. Mawla, and J. Benson, "Reachability analysis for attributes in ABAC with group hierarchy," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 841–858, 2023.

[22] G. Fragkos, J. Johnson, and E. E. Tsiropoulou, "Dynamic role-based access control policy for smart grid applications: an offline deep reinforcement learning approach," *IEEE Transactions on Human-Machine Systems*, vol. 52, no. 4, pp. 761–773, Aug. 2022.

[23] P. G. Neumann, "Security and privacy," *Computer Science Handbook*, Taylor & Francis, pp 77–81, Florida, NY, USA, 2004.

[24] X. Li and X. Zhao, "Survey on access control model in cloud computing environment," in *Proceedings of the 2013 International Conference on Cloud Computing and Big Data*, pp. 340–345, CLOUDCOM-ASIA, Fuzhou, China, December 2013.

[25] M. Ahmed and M. Ashraf Hossain, "Cloud computing and security issues in the cloud," *International journal of Network Security & Its Applications*, vol. 6, no. 1, pp. 25–36, 2014.

[26] A. Majumder, S. Namasudra, and S. Nath, "Continued rise of the cloud," *Computer Communications and Networks*, Springer, Verlag London, 2014.

[27] R. K. Aluvalu and L. Muddana, "A survey on access control models in cloud computing," *Advances in Intelligent Systems and Computing*, vol. 337, pp. 653–664, 2015.

[28] I. Indu, P. R. Anand, and V. Bhaskar, "Identity and access management in cloud environment: mechanisms and challenges," *Engineering Science and Technology, an International Journal*, vol. 21, no. 4, pp. 574–588, 2018.

[29] F. Sifou, A. Hammouch, and A. Kartit, "Innovations in smart cities and applications," *Proceedings of the 2nd Mediterranean Symposium on Smart City Applications*, vol. 1, pp. 255–264, 2018.

[30] G. Karataş and A. Akbulut, "Survey on access control mechanisms in cloud computing," *Journal of Cyber Security and Mobility*, vol. 7, no. 3, pp. 1–36, 2018.

[31] R. El Sibai, N. Gemayel, J. Bou Abdo, and J. Demerjian, "A survey on access control mechanisms for cloud computing," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, 2020.

[32] B. Cha, J. Seo, and J. Kim, "Design of attribute-based access control in cloud computing environment," *Lecture Notes in Electrical Engineering*, vol. 120, pp. 41–50, 2012.

[33] G. Lin, Y. Bie, and M. Lei, "Trust based access control policy in multi-domain of cloud computing," *Journal of Computers*, vol. 8, no. 5, pp. 1357–1365, 2013.

[34] L. Mohan and M. S. Elayidom, "Fine grained access control and revocation for secure cloud environment-a polynomial based approach," *Procedia Computer Science*, vol. 46, pp. 719–724, 2015.

[35] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: a survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–41, 2020.

[36] M. Habiba, M. R. Islam, A. B. M. S. Ali, and M. Z. Islam, "A new approach to access control in cloud," *Arabian Journal for Science and Engineering*, vol. 41, no. 3, pp. 1015–1030, 2016.

[37] J. C. Kimmell, M. Abdelsalam, and M. Gupta, "Analyzing machine learning approaches for online malware detection in cloud," in *Proceedigs of the 2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 189–196, Irvine, CA, USA, August 2021.

[38] A. Saini, Q. Zhu, N. Singh, Y. Xiang, L. Gao, and Y. Zhang, "A smart-contract-based access control framework for cloud smart healthcare system," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5914–5925, 2021.

[39] J. Xu, Y. Yu, Q. Meng, Q. Wu, and F. Zhou, "Role-based access control model for cloud storage using identity-based cryptosystem," *Mobile Networks and Applications*, vol. 26, no. 4, pp. 1475–1492, 2021.

[40] S. Almutairi, N. Alghanmi, and M. M. Monowar, "Survey of centralized and decentralized access control models in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 2, 2021.

[41] E. A. Shammar, A. T. Zahary, and A. A. Al Shargabi, "An attribute-based access control model for internet of things using hyperledger fabric blockchain," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 6926408, 25 pages, 2022.

[42] M. C. Gaudel, "Formal specifications and software testing, a fruitful convergence," in *Formal Methods. FM 2019 International Workshops. FM 2019*, Springer, Cham, 2020.

[43] M. Soavi, N. Zeni, J. Mylopoulos, and L. Mich, "From legal contracts to formal specifications: a systematic literature review," *SN COMPUT. SCI.,*vol. 3, no. 5, p. 345, 2022.

[44] D. Cansell and D. Méry, "Foundations of the B method," *Computing and Informatics*, vol. 22, no. 3–4, pp. 221–256, 2003.

[45] V. S. Alagar and K. Periyasamy, *Specification of Software Systems*, Springer, UK, 2011.

[46] J. Woodcock and J. Davies, "Using Z. Structure," 1996, http://www.usingz.com/.

[47] G. O'Regan, "Z formal specification language," in *Mathematics in Computing*, Springer, Cham, 2020.

[48] D. Power, M. Slaymaker, and A. Simpson, "On formalizing and normalizing role-based access control systems," *The Computer Journal*, vol. 52, no. 3, pp. 305–325, 2009.

[49] P. Saratha, G. V. Uma, and B. Santhosh, "Formal specification for online food ordering system using z language," in *Proceedings2017 2nd International Conference on Recent Trends and Challenges in Computational Models, ICRTCCM*, pp. 343–348, Tindivanam, India, February 2017.

[50] M. M. Noaman, I. M. Alsmadi, and A. S. Jaradat, "The specifications of E-Commerce Secure System using Z language. II(III)," in *Proceedings of the 2013 Conference on Innovations in Computing and Engineering Machinery Conference*, Amman, Jordan, April 2013.

[51] M. U. Siregar and J. Derrick, "A scanner and parser for Z specifications," *IJID (International Journal on Informatics for Development)*, vol. 7, no. 1, pp. 13–20, 2018.

[52] L. Lamport and F. B. Schneider, "The``Hoare logic"of CSP, and all that," *ACM Transactions on Programming Languages and Systems*, vol. 6, no. 2, pp. 281–296, 1984.

[53] D. Jackson, "Alloy: a language and tool for exploring software designs," *Communications of the ACM*, vol. 62, no. 9, pp. 66–76, 2019.

[54] P. Tolmach, Y. Li, S. W. Lin, Y. Liu, and Z. Li, "A survey of smart contract formal specification and verification," *ACM Computing Surveys*, vol. 54, no. 7, pp. 1–38, 2021.