Evaluating A Security Framework for Access Control in SaaS Systems

MSc Research Project
MSc in Cybersecurity

Godwin Akinbode
X20259433

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Godwin Akinbode
Student ID:	X20259433
Programme:	Cyber Security
Year:	2022
Module:	MSc Research Project
Supervisor:	Michael Pantridge
Submission Due Date:	15th August 2022
Project Title:	Evaluating a Security Framework for Access Control in SaaS Systems
Word Count:	6122
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating A Security Framework for Access Control in SaaS Systems

Godwin Akinbode
X20259433
MSc Project in Cyber Security

Abstract

Software-as-a-Service (SaaS) is a new approach to software development that is distinguished by its multitenancy architecture and ability to provide flexible customization to individual tenants. Multi-tenancy is at the core of SaaS as it allows users (tenants) and providers the opportunity to fully exploit the full potential of cloud computing. However, due to multi-tenancy, access control has become a critical component in the development of SaaS systems, particularly for limiting cross-tenant access to resources. It's one of the most important aspects of computer security and the ability to manage access to a SaaS system is a critical security problem that must be addressed. This research describes an improved version of Hierarchical Access Control Model (HRBAC) model called SAMRDT (SaaS Authorization management Resource Directed tree) which is based on resource directed tree. It combines the benefits of the ABAC and RBAC access control models and adds effective authorization management to the SaaS access control architecture. As part of our research work, a prototype system based on the model was implemented to evaluate the performance. We also specified a set of metrics that will be used to quantitively and objectively evaluate the performance of SAMRDT model for authorization management and permission filtering in SaaS.

1 Introduction

In this day and age of cloud computing, there has been an exponential increase in the use of multi-tenant applications, most especially SaaS (Software as a Service) applications, which are now widely regarded as the next generation of Internet applications. According to a report published by Businesswire, the international software as a service (SaaS) market is expected to increase at a compound annual growth rate (CAGR) of 20.8 percent from $225.6 billion in 2020 to $272.49 billion in 2021; at a CAGR of 12.5 percent, the market is estimated to reach $436.9 billion in 2025 (Laura Wood, n.d.). Software as a Service (SaaS) providers provide software as a service over the Internet, removing the requirement for users to download and execute the software or applications on their own devices/servers (D. Li et al., 2010). Resource sharing is one of the foundations of cloud computing as resource sharing at various software levels enables cloud service providers exploit the benefits of cloud computing at a very low cost which results in financial savings for the user. Multi-tenancy has risen to prominence as a technology that can help organizations in effectively sharing cloud resources while optimizing their sales profit and

lowering the cost of hosting the application, software development and its maintenance (Abdul et al., 2018)
Multi-tenancy is a well-known strategy for lowering total cost of ownership. In a multi-tenancy model, tenants' data is stored in the same public cloud and is controlled according to the user by the tagging of resources owned by the user (Kanade & Manza, 2019). Multi-tenancy in a SaaS application is an architectural pattern that comprises numerous tenants (customers) sharing resources at various layers of the application. This could include tenants sharing the same user interface components, business logic, and data layer, but each tenant only has access to functions or components and data belonging to their own business segment (Abdul et al., 2018). This technology allows SaaS providers run an instance of their software application on a database instance and give numerous tenants access to the application. Every customer is a tenant who has the right to modify the user interface or business rules as he sees fit with no right to modify the application code. This is the foundation of multi-tenancy in the SaaS platform.

Fig. 1 illustrates the difference between a single-tenant architecture and a multi-tenant architecture and the benefits of choosing multi-tenant architecture for SaaS applications. Fig. 1 shows that a single software application and its infrastructure and database is being used to serve multiple. Every client will have access to the same software application as well as a single database.

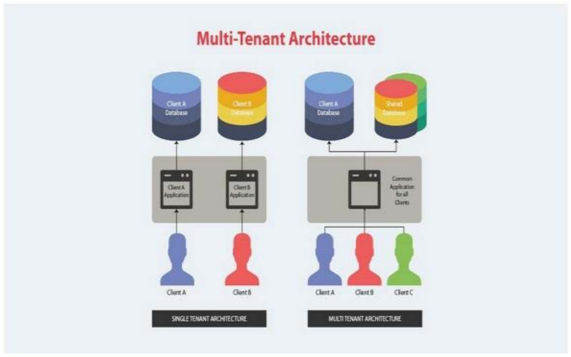


Fig. 1. A Single and Multi-tenant architecture

One of the many advantages of a multi-tenant application is its capability to support various functionalities which may be similar or different and its ability to scale horizontally (Abdul et al., 2018); however multi-tenancy introduces some security risks and one of the most critical risks of multi-tenancy in a SaaS architecture is access of data by other tenants. (Solanki et al., 2016) implied that security is a big problem with SaaS platforms because of the centralized hosting of client data software; they also mentioned that SaaS platforms insusceptible to some common security issues. However, there are other security challenges unique to SaaS platforms, which multi-tenancy access control is one of.
An access control model focuses primarily on access control and management, yet; further research on the SaaS access control paradigm emphases solely on access control, discounting the difficulty of multitenant access management. It’s imperative that a complete SaaS security access control solution is implemented into the SaaS platform to ensure that organizations are willing to entrust their company and data to SaaS providers for processing and storage (Guo, 2018). In SaaS environments, the tenant administrator normally assigns the users' permissions to the tenant's resources individually to certify the discretion of tenant data; if permission

assignment is overly complicated, it will have a direct impact on access control accuracy and SaaS promotion (Wang, Wang and Zhang, 2020). Existing access control frameworks have been presented, yet the majority of them are unable to deliver simple, flexible and fine-grained access control specifically for SaaS environment.
(Wang et al., 2020) proposed a novel authorization management model called the SAMRDT (SaaS Authorization Management using Resource Directed tree) model which effectively controls the workload of authorization management while simultaneously ensuring that session permissions are fine-grained and dynamically restricted.

But little work has been done on evaluating the proposed model as only formal methods for the validation of the novel SAMRDT model involving mathematical formula and proofs have been advocated. Therefore, since a research work is incremental, the contribution of this paper is to perform a simulation testing and evaluation of the SAMRDT model to determine the efficiency and performance tradeoffs of the model.

1.1 Research Question

How can the use of Resource directed tree and a hybrid of ABAC (Attribute based Access control) and RBAC (Role based Access Control) models improve the access control and management of SaaS application resources?

The remainder of the paper is organized as follows: The literature review is discussed in Section II, and the research methods and specifications which includes details about the analysis of model are discussed in section III. Section IV includes the design specification. Section V and VIII provides information about the implementation and evaluation of proposed model. Finally, Section concludes the paper and discusses future work.

2 Related Works

In this section, we discuss various significant works and existing models for the access control of resources and the access control of resources in a SaaS platform that have been described in the literature. This paper discusses various aspects of access control models, including the traditional access control models, RBAC-like access control models, H-RBAC access control model and the SAMRDT access model. The study also highlights the limitations of previous research that influenced the development of this project.

2.1 Traditional Access Control Models

The three most common access control models are Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) (Lang et al., 2008). The DAC model is highly flexible but lacks adequate security, and the management of authorization is complicated. Although the Mandatory access control (MAC) is a more secure type of access control, it is quite inflexible. Thus, RBAC is the widely known and used a method of access control (Xiong, 2015). RBAC is a powerful and versatile access control model that falls between the DAC and MAC models, and it excels at representing the semantic meaning of the relationship between user. (Denim et al., 2003). It has the ability to not only represent "responsibility" in complicated social systems, but also improves authorization management and reduces administrator workload.

The RBAC model was originally introduced in 1992 by David Ferraiolo and Rick Kuhn and this model introduces the “role” concept between “user” and “permission”. In this model, each user is associated with one or more roles, and each role may be associated with one or more permissions and the roles are created or deleted according to business needs (David & Richard, 1992). RBAC96 was proposed where the RBAC model was modified by adding a role hierarchy and assigning constraints to the model (R. S. Sandhu et al., 1996) and an (R. Sandhu et al., 1999.) presented a RBAC97 model in 1997 which was an improvement on RBAC 96 allowing for more administrative ease and scalability, particularly in decentralizing administrative responsibility but still lacked organization. A new model RBAC02 was later proposed which improves on the RBAC97 by adding the concept of ‘organization’ to the core characteristics of the RBAC97 model (Sejong & Ravi, 2002). (Ma & Li, 2010) conducted some study on SaaS system’s admission control, operation control, and data access control, but they did not conduct extensive research on access control and did not propose a practical implementation of the access control module. However; despite the model upgrades, RBAC still had some limitations like its inability to perform dynamic and fine-grained authorization. Traditional access control models such as DAC, MAC, and RBAC are identity-based, with subjects of interest being identified either directly or through roles assigned to them. These models are only appropriate for centralized systems that do not change and have a small number of users and services, hence there are several problems to using centralized RBAC models to control access in a multi-tenancy environment since the roles, permissions, and users are constantly changing.

The ARBAC97 model was proposed which introduces URA97 (User-Role Assignment), PRA97 (Permission-Role Assignment), and RRA97 (Role-Role Assignment) to solve the static characteristic of RBAC (Li et al., 2011). This model offers administrative ease and scalability, specifically in decentralising administrative responsibility and roles. Later, when the ARBAC97 model had been used for a few years to define user-role assignments, the ARBAC02 model was used to define the role-permission assignments for users of the ARBAC02 model. Later, when the ARBAC97 model had been used for a few years to define userrole assignments, the ARBAC02 model was presented to define the role-permission assignments for users of the ARBAC02 model. ARBAC02 improves on ARBAC97 by adding the concept of "organisation" to the core characteristics of ARBAC97
(Sejong Oh & Ravi Sandhu. 2002). A tenant-based access control model T-Arbac was later proposed by adding tenant into ARBAC model and separating the functions of system administrators and tenant administrators (Cao et al., 2013) ABAC (Attribute-based Access Control) was presented which turned out to be an even more complex solution in terms of strategic review (Zhou & Ren, 2018). The policies in the ABAC model are more semantically expressive than those in the RBAC model, allowing for more efficient service composition (Jiang & Zhang, 2012); therefore, some researchers have steered towards combining the advantages of both ABAC and RBAC models (Geetha & Anbarasi, 2018; Kamath et al., 2006; Penelova, 2021; Ravidas et al., 2019; Varadharajan & Rai, n.d.) which has solved some access control issues but failed at improving the access control in a SaaS multi-tenancy architecture. Most of the literature on access control model focus on three categories which are namely access control of role-centric access control (Al-Kahtani & Sandhu, 2002), dynamic roles (Jin et al., 2012; Jin et al., 2012) and attribute-centric access control (Rajpoot et al., 2015, Jin et al., 2012, Xiong et al., 2016).

2.2 RBAC-Like Models for Multi-tenant isolation

(X. Y. Li et al., 2010) introduced the RBAC model to SaaS systems and the following issues were discovered: cross-level management, role name conflicts and the isomerism of the tenant' s access control. More recent research also identified the same issues (Zuo et al., 2017). They proposed the S-RBAC model which simplifies the authorization management by dividing access control into tenant-level and system-level but failed at role hierarchy. A Hierarchical Access Control Model (H-RBAC) model which solved the delegation and time-constraints issue in the S-RBAC model was presented (D. Li et al., 2011).
T-Arbac was presented as a tenant-based access control approach which includes 'tenants' into the ARBAC model and separates the functions of system administrators and tenant administrators (Cao et al., 2013). In this model, system resources are divided into Sub-resource pools and each tenant is assigned a specific pool which isolates them. The TMS-ARBAC model also places an emphasis on the importance of the isolation of tenants in SaaS environment (Zuo et al., 2017) by introducing the AA-tree and Autonomous areas (AA) to depict tenants' autonomy, including their isolation and sharing ties but more research needs to done by focusing on the scalability and distributing features of MTA and STA SaaS systems to increase the efficiency of authorization and authentication. Both (Calero et al., 2010; Maenhaut et al., 2015) discovered that the issue is the establishment of a role hierarchy among tenants. Unlike within an organization, where a natural hierarchy of roles exists, tenants may not have such hierarchical relationships. Furthermore, unlike the role hierarchy model, access rights among tenants cannot always be coerced into an acyclic partial ordering. considers a primitive capability-list based model for cross-tenant access right specification. This model may be useful if only a few cross-tenant access privileges are to be given; otherwise, assignment and management of cross-tenant privileges can be complex. The MT-RBAC model is a multi-tenant role-based access control model that allows tenants' truster-trustee relationships to be established first and each tenant can later have either public or private roles.

MT-RBAC is comparable to the role mapping method (Kamath et al., 2006) but it protects the integrity of tenants' role hierarchies. (Tang et al., 2013). (Solanki et al., 2016) introduced a new and enhanced model 'SaaS-AIFC' for SaaS access and information flow control but the access control is role-based which makes it unsuitable for tenants in a SaaS due to its interoperation flaws in cross-tenant access control.

2.3 H-RBAC Model

The H-RBAC model provides a solution to the access control problem in SaaS systems by improving the tenant-based access control model (D. Li et al., 2011). This model divides access control into tenant-level and system-level access control. The tenant-level access control model relies on the organization-based access control model, while the system-level access control model relies on the administrative role-based access control model which is based on the ARBAC97 model. It provides a mechanism for distinguishing admin roles from general roles. H-RBAC provides a structure that is adaptable to hierarchical responsibility sections and authority management, high scalability, and ability to meet the needs of a variety of tenants in a heterogeneous access control system, tenants' self-government; without reinforcing the role of multiple-level hierarchies. However, the model is faced with security, efficiency, and permission constraints. As a result, after examining the benefits and drawbacks of existing access control models, this study assesses a SAMRDT access control model for SaaS systems.

2.4 SAMRDT Model

The SAMRDT (SaaS Authorization management Resource Directed tree) is an access control model that tries to solve the limitations of the H-RBAC model (which retains the RBAC model characteristics) while complying to the minimum authorization and separation of duties principles (Wang et al., 2020). This model introduces the use of a resource directed trees in which the resources of a SaaS platform are arranged by business logic. It introduces the resource directed tree between roles and resources of the H-RBAC model. Role permission assignments based on the resource directed tree could lessen the effort of authorization management due to the inherent business logic connecting resources in a tree. An attribute-based permission reduction strategy that combines the ABAC and RBAC is introduced in response to fine-grained authorization management. Because role permissions in the model are assigned based on the resource directed tree, the tree also defines permission filtering policy is also defined on the tree.

The SAMRDT architecture is depicted in Figure 2 shows the RDT which stands for resource directed tree and the resource permission tree is the resource directed tree after authorization occurs. The SAMRDT model uses TENANT, USER, SESSION, ROLE, RES, and PERMS to represent the sets of tenants, users, sessions, roles, resources, and permissions, and follows the HRBAC model's definition of tenant, user, session, role, resource, permission, and other aspects. In the H-RBAC model, resource directed trees are introduced between roles and resources, in which the resources of a SaaS platform are organized by business logic. As a result, the traditional permission assignment method of 'role– resource– permission' has evolved into 'role–'resource directed tree' –resource-permission' without jeopardizing the HRBAC model's security. The resource directed trees in which the resources of SaaS platform are organized by business logic, are introduced between roles and resources in the H-RBAC model, the traditional permission assignment method of ‘role– resource–permission’ has evolved into ‘role– ‘resource directed tree’ –resource-permission’ without destroying the security of the HRBAC model

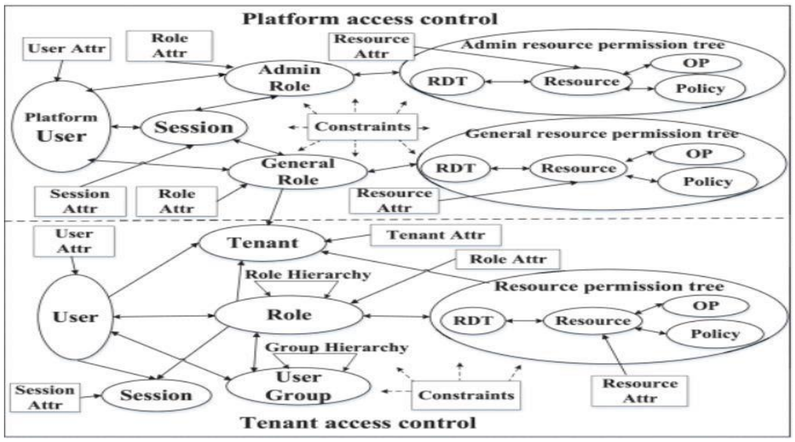


Figure 2: SAMRDT Model

3 Research Methodology

The research methodology section outlines a description of the methods that make up the prototype system for effectively assigning permission to the tenants of a SaaS application. The purpose of the model is to enhance the authorization management of the resources of SaaS application by combining the advantages of the RBAC and the ABAC access control methods. The models are combined to take advantage of both of their benefits, as doing so not only ensures that session permissions are tightly and dynamically controlled but also significantly reduces the workload associated with authorization management. The model consists of three main methods:

- Authorization of user's access to resources based on their role
- Attribute-based permission filtering or reduction which furthers constraints the available resources allocated to a user role,
- Resource directed tree.

3.1 Role-based Authorization

System access is limited by the mechanism known as role-based access control (RBAC), also referred to as role-based security. To grant access to authorized users, it involves setting privileges and permissions. Role-based access control is a common practice in large organizations to give employees different levels of access based on their roles and responsibilities. In the role-based access control approach, users are assigned to predefined roles that have a certain set of privileges attached to them. For instance, a user allocated the role of Manager will have access to diverse set of resources than a user assigned the function of Analyst. In this approach, access is expressly predetermined by the object owner when deciding the permission associated with each role and implicitly predetermined by the person who assigns the roles to each individual. Least-privilege can be implemented with RBAC; however, many SaaS apps make it challenging to do so. Without fine-grained control over admin access, permissions are either extremely excessive or a hurdle to productivity. RBAC must enforce least privilege by only granting access to resources that people require to perform their jobs by applying stringent access restrictions to sensitive data, systems, and applications. Unfortunately, least privilege is complicated with SaaS management since different SaaS apps define user roles differently and have different levels of granularity. Each role of the prototype authorization management system has specific resources allocated to them. The users of the system will have permission to access specific resources of the application based on their specific role.

3.2 Attribute-based permission reduction

Although the workload of permission management is significantly reduced by the role permission assignment based on the resource directed tree, it is still challenging to implement fine-grained and dynamic permission assignment. In order to better support the principles of least privilege and separation of duties and increase the security of access control, the attribute-based permission filtering policy is defined to fine-grained and dynamically constrain the available permissions of the session role (Wang et al., 2020). Some permissions that do not meet the attribute constraints among the available permissions of the session role are deleted. The authorization management system

3.3 Resource directed tree

A resource directed tree is a directed acyclic SaaS platform resources arranged according to business logic. The business logic provided by the SaaS program code or defined by the administrators determines the relationships between resource nodes (Wang et al., 2020).

4 Design Specification

This section describes the proposed system's flow and emphasizes how the various system components were designed. The proposed model is divided into three main sections, as was discussed in the previous section, and is depicted in Figure 1. These are the role-based authorization, attribute-based permission reduction and the use of resource directed tree to assign role permission assignments which reduces the workload of authorization management.

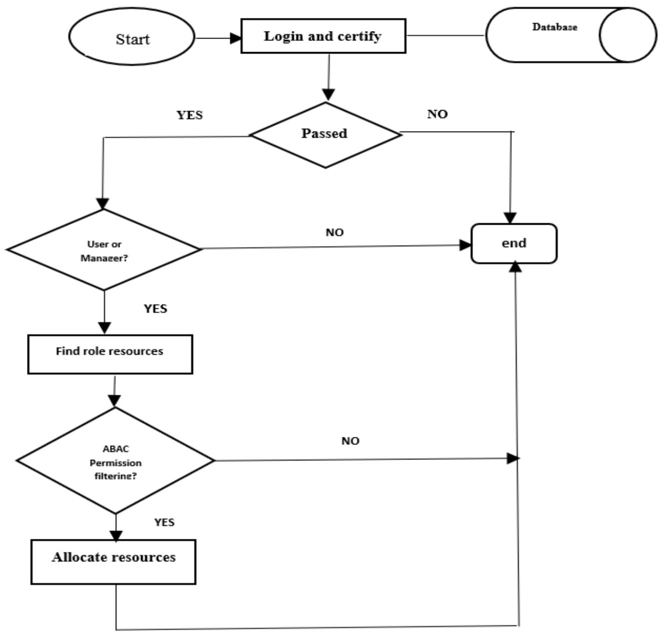


Figure 3: Flow diagram of the Application

In the SaaS-based authorization management system, the user must first enter their email address and password on the SaaS management console. The console then verifies the user's role and tenancy after confirming the email address and password. Third, the console obtains the list of resources and their access permissions for the user, and it then returns the user's access permissions. The UML (unified modeling language) sequence diagram is displayed below:

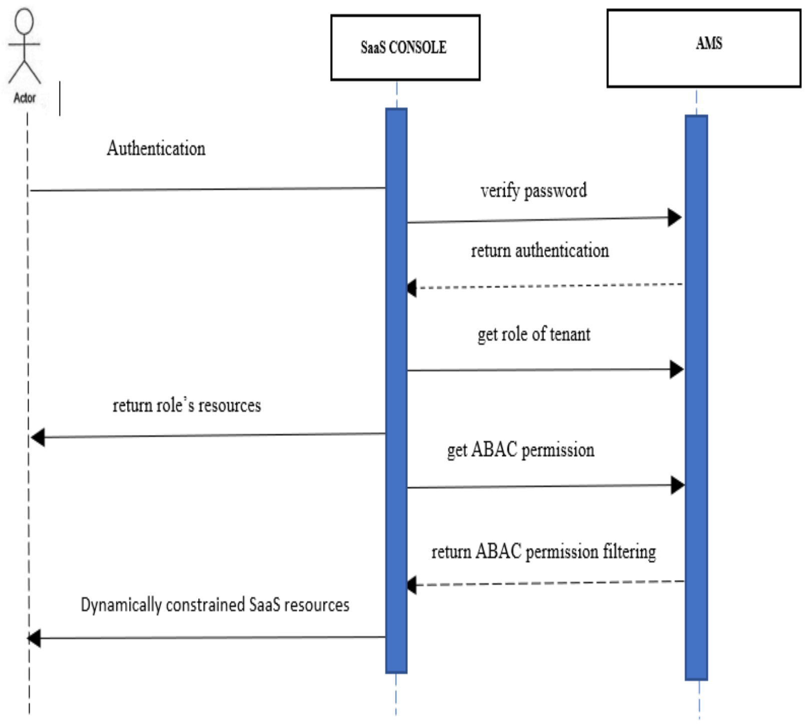


Figure 4: Sequence diagram of the authorization system

4.1 Proposed Method Algorithm

Step 0: Using secondary datasets provided by the researcher, the system accepts user email and password for verification.
Step 1: After authentication, the system gets tenant and determines if the tenant is either a manager role or an employee role.
Step 2: The systems get the resource directed tree associated with the specific role
Step 3: Get available resources according to the resource directed tree
Step 4: Uses ABAC permission filtering to further constrain the available resources in **Step 3** according to tenant's attributes
Step 5: Provides permission to remaining resource**.**

5. Implementation

In this section, we describe the prototype implementation of a SaaS authorization management system. The implementation of the proposed solution has been developed by building a prototype application to evaluate how the model will manage authorization in a real-world situation. This paper proposes a resource authorization management model that is based on a hybrid access control, implemented below by combining the RBAC model and ABAC model to constraint resources available to users and enhance its security. The proposed system uses both static and dynamic permission allocation.

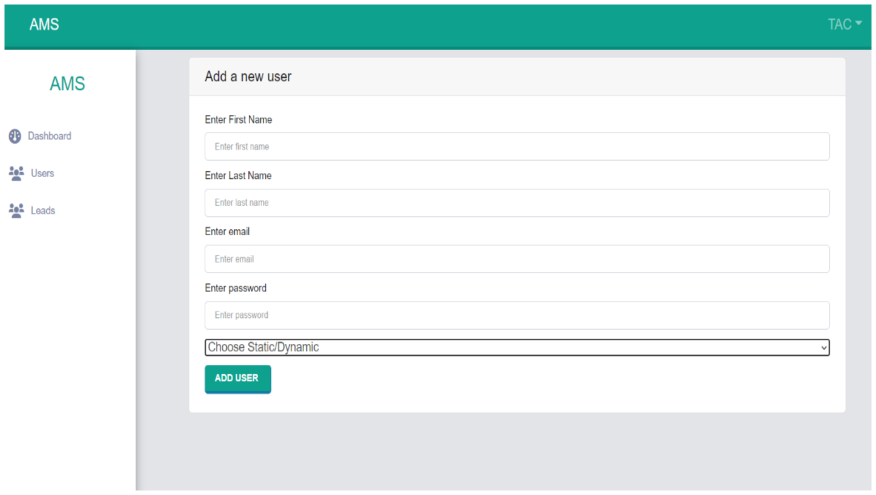
The prototype application is built to showcase how the various tenants of a SaaS application are allocated access to the resources of an application. The first page is the login page where users of a SaaS application input their credentials i.e., email address and password which is validated against a backend database to ensure the users are who they claim to be hereby enhancing secure authentication into the application.

As an example, we describe an application of SAMRDT model for an authorization system. The system allows users to create authenticate themselves using secure credentials. Registered users can now perform actions and have access to specific part of the application based on their given role and the permissions assigned to them based on their attributes. A manager will have access to edit articles created by users. The administrator has access to all sections, including user administration. The prototype system has 3 roles, over 8 user-role assignments and a total of 20 users who will have different access to the resources of the application based on their role and based on the available permission to the application's resources.

Users U: {John, Sophy, Carter, Alice}
Roles R: {Manager, User, Administrator}
Permissions P: {"view user", "update user", "add user", "delete user", "view lead", "delete lead", "update lead", "dashboard",}.

5.1 Initial role-based authorization using static permissions

The initial step is to add new tenants to the prototype application using the add a new user feature of the application. This action can only be taken by a tenant administrator or a manager or a user who has been given the permission to access such resource.



Add a new user

Enter First Name

Enter first name

Enter Last Name

Enter last name

Enter email

Enter email

Enter password

Enter password

Choose Static/Dynamic

ADD USER

Figure 5: Adding a new user

Once the user's details have been inputted, the administrator choses a secure password for the user which will later be forwarded to the user. As mentioned previously, the administrator can allocate static permissions to the new user which allows the user have access to specific resources of the SaaS application.

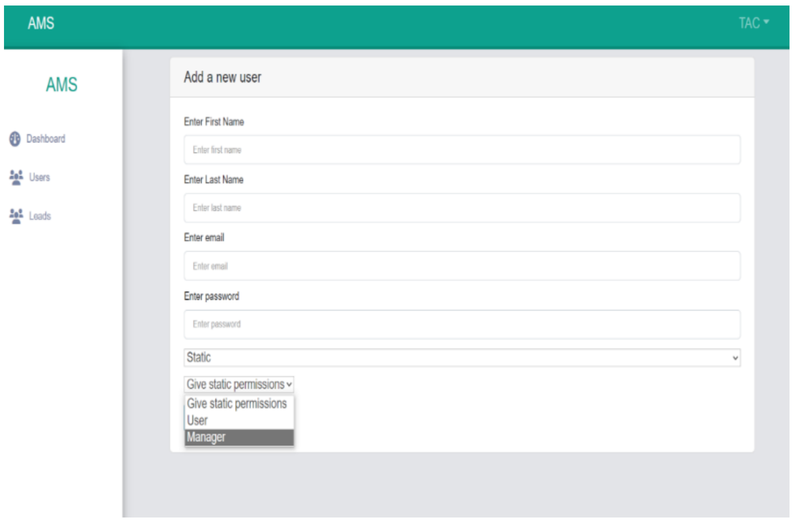


Figure 6: Static permission

Figure 6 highlights the static allocation of permission to the new user; the administrator has the options of allocating permission to the resources of the application based on the role of the new user. If the administrator allocates the User role to the user, he will have access to specific resources of the application like the leads/data section, while the Manager role has access to the user's section and can also perform actions like creating, updating, and deleting users etc.

5.2 Implementing the Hybrid Approach

Now that the initial role-based permission allocation is completed, the administrator can decide to implement the dynamic permission which further constrains or expands the resources available to a particular user role hereby prompting least privilege in the application. In this prototype application, the application also uses a dynamic method of permission allocation.

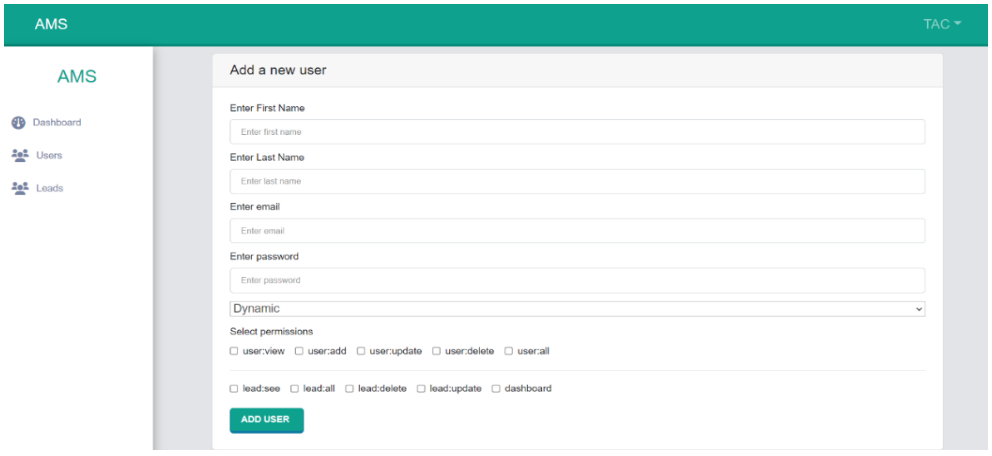


Figure 7: Dynamic permission

Figure 7 demonstrates the dynamic allocation of resources to the tenants/users of a SaaS application. While some dynamic role implementation may allow the front-end attribute engine to fully determine the user's role, others may limit its use to selecting from a predetermined list of approved roles. This prototype system uses a front-end attribute engine to determine the user’s role and determine the resources available to the user irrespective of the allocated role. In SaaS, the tenant administrator typically independently assigns the users' access to the resources to ensure the privacy of tenant data.

6 Evaluation

The evaluation section of this paper portrays the experiment performed on the prototype authorization management system based on the SAMRDT model. The proposed method comprises of RBAC, and ABAC models and the introduction of a resource directed tree, while the alternative method comprises of only the RBAC model. The reason behind the selection of RBAC & ABAC models is that the models are combined to take advantage of both of their benefits, as doing so not only ensures that session permissions are tightly and dynamically controlled but also significantly reduces the workload associated with authorization management. The idea behind this evaluation is to give the appropriate individuals access ensuring that only permitted individuals can view data/resources in the prototype SaaS application.

6.1 System Access control test

We executed a test with 16 participants, this time as users, to find out
how efficient our proposed system is. To do this, we made use of use of secondary dataset(s) created by the researcher, where the 16 users were added to the prototype system and assigned various static and dynamic permissions using the authorization management system. The login details of these users were provided to some real-life participants and the resources of the application which were available to them were recorded.

The resources the participants had access to were compared to the resources allocated to them. The table below illustrates the 16 participants and the type of permission allocated to them where D= Dynamic and S = Static permission. We compared the permissions they could access to the permissions allocated to them by the administrator. The total number of permissions that could be assigned to the user is 10 permissions and only the Manager/Administrator role are assigned those statically.

Users	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Permission Type	D	S	D	S	S	S	D	D	S	D	D	S	D	D	S	S
Permissions Allocated	6	8	3	8	8	1	2	2	1	6	6	8	2	4	1	1
Resources Accessed	4	8	3	8	8	1	2	2	1	6	6	8	2	4	1	1

Table 1: illustration of permission allocated & resources accessed

6.2 Results

The table 2 shows the estimated results from the evaluation. The performance metrics used to evaluate the proposed model are similar to the ones used by the researchers in (Le et al., 2014)

Access Control Model	**Access Level**	**Accuracy**	**Specificity**	**Management complexity**
SAMRDT model	High	High	High	Easy
Traditional RBAC	Low	Medium	High	Difficult

Table 2: Result of evaluation done

6.3 System security check

One of the most crucial features of today's multi-tenant applications is the impact on security. To make sure the application can withstand threats in the real world, a security audit at the code level and application level must be carried out. To quickly verify this, we ran tests using OWASP's ZAP and Nessus.

6.4 Discussion

This research has demonstrated that the hybrid model performs better than a single access control model when compared to itself. Metrics such as access level, accuracy, specificity, and management complexity were analyzed in this evaluation. The result in Table.2. shows that the SAMRDT model has an access level of ‘High’ in comparison to the RBAC model because the users assigned permission to specific resources using the static permission will experience role-permission explosion and role-explosion problems. In this research work, a hybrid access model proposed, the SAMRDT hybrid model giving a “high” access level, accuracy, and a management complexity of “Easy” while the traditional RBAC model achieved a specificity of ‘high’ due to its role-permission methodology.

7 Conclusion and Future Work

The main aim of this research was to improve the effective management and assignment of the resources of a SaaS application to multiple users depending on the permission given to them. This research proposes an access control model that combines the advantages of RBAC, and ABAC models and determines how the use of Resource directed tree would enhance the access control and management of SaaS application resources. The results gotten from our evaluation demonstrates that Traditional RBAC shows promise if the motivation is for specificity and accuracy and SAMRDT show promise if the goal is ease of management complexity, access level, specificity, and accuracy. In conclusion, the SAMRDT model outperforms the RBAC approach in terms of security and management complexity. A limitation of this study is that the application had insufficient number of resources to truly capture diverse real-world application

In our future works, we aim to complete the implementation of the proposed model. Some features of the SAMRDT model like permission derivation weren’t implemented, also the datasets used to evaluate the system were generated by the researcher and stored in the database manually, the use of publicly available dataset is recommended. Furthermore, the authorization management system lacks proper and secure authentication as the focus of this research was solely on the authorization of users. In terms of the resource directed tree, more research needs to be carried out to determine the drawbacks of this approach.

References

Abdul, A. O., Bass, J., Ghavimi, H., Macrae, N., & Adam, P. (2018). *Multi-tenancy Design Patterns in SaaS Applications: A Performance Evaluation Case Study*.

Access Control Policy Testing | CSRC. (n.d.). Retrieved April 10, 2022, from https://csrc.nist.gov/projects/access-control-policy-tool

Al-Kahtani, M. A., & Sandhu, R. (2002). A model for attribute-based user-role assignment. *Proceedings - Annual Computer Security Applications Conference, ACSAC*, *2002-January*, 353–362. https://doi.org/10.1109/CSAC.2002.1176307

Calero, J. M. A., Edwards, N., Kirschnick, J., Wilcock, L., & Wray, M. (2010). Toward a multi-tenancy authorization system for cloud services. *IEEE Security and Privacy*, *8*(6), 48–55. https://doi.org/10.1109/MSP.2010.194

Geetha, N., & Anbarasi, M. S. (2018). Role and attribute based access control model for web service composition in cloud environment. *ICCIDS 2017 - International Conference on Computational Intelligence in Data Science, Proceedings*, *2018-January*, 1–4. https://doi.org/10.1109/ICCIDS.2017.8272642

Jiang, H., & Zhang, H. (2012). Access control model for composite web services. *International Conference on Communication Technology Proceedings, ICCT*, 684–688. https://doi.org/10.1109/ICCT.2012.6511292

Le, X. H., Doll, T., Barbosu, M., Luque, A., & Wang, D. (2014). Evaluation of an Enhanced Role-Based Access Control model to manage information access in collaborative processes for a statewide clinical education program. *Journal of Biomedical Informatics*, *50*, 184–195. https://doi.org/10.1016/J.JBI.2013.11.007

Kamath, A., Liscano, R., & el Saddik, A. (2006). *User-credential based role mapping in multi-domain environment*. 1. https://doi.org/10.1145/1501434.1501507

Kanade, S., & Manza, R. (2019). A Comprehensive Study on Multi Tenancy in SAAS Applications. *International Journal of Computer Applications*, *181*(44), 25–27. https://doi.org/10.5120/IJCA2019918531

Lang, B., Foster, I., Siebenlist, F., Ananthakrishnan, R., & Freeman, T. (2008). A Flexible Attribute Based Access Control Method for Grid Computing. *Journal of Grid Computing 2008 7:2*, *7*(2), 169–180. https://doi.org/10.1007/S10723-008-9112-1

Laura Wood. (n.d.). *Software as a service (SaaS) Global Market Report 2021: Focus on Customer Relationship Management; Enterprise Resource Planning; Human Resource; Manufacturing and Operations; Supply Chain Management - ResearchAndMarkets.com | Business Wire*. Retrieved April 6, 2022, from https://www.businesswire.com/news/home/20210820005229/en/Software-as-a-service-SaaS-Global-Market-Report-2021-Focus-on-Customer-Relationship-Management-Enterprise-Resource-Planning-Human-Resource-Manufacturing-and-Operations-Supply-Chain-Management---ResearchAndMarkets.com

Li, D., Liu, C., & Liu, B. (2011). H-RBAC: A Hierarchical Access Control Model for SaaS Systems. *International Journal of Modern Education and Computer Science*, *3*(5), 47–53. https://doi.org/10.5815/IJMECS.2011.05.07

Li, D., Liu, C., Wei, Q., Liu, Z., & Liu, B. (2010). RBAC-based access control for SaaS systems. *2nd International Conference on Information Engineering and Computer Science - Proceedings, ICIECS 2010*. https://doi.org/10.1109/ICIECS.2010.5678213

Li, X. Y., Shi, Y., Yu-Guo, & Ma, W. (2010). Multi-tenancy based access control in cloud. *2010 International Conference on Computational Intelligence and Software Engineering, CiSE 2010*. https://doi.org/10.1109/CISE.2010.5677061

Maenhaut, P. J., Moens, H., Ongenae, V., & de Turck, F. (2015). Design and evaluation of a hierarchical multi-tenant data management framework for cloud applications. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, 1208–1213. https://doi.org/10.1109/INM.2015.7140468

Penelova, M. (2021). Hybrid Role and Attribute Based Access Control Applied in Information Systems. *Cybernetics and Information Technologies*, *21*(3), 85–96. https://doi.org/10.2478/CAIT-2021-0031

Ravidas, S., Lekidis, A., Paci, F., & Zannone, N. (2019). Access control in Internet-of-Things: A survey. *Journal of Network and Computer Applications*, *144*, 79–101. https://doi.org/10.1016/j.jnca.2019.06.017

Sandhu, R., Bhamidipati, V., Co, E., Ganta, S., & Youman, C. (n.d.). *'The ARBAC97 Model for Role-Based Administration of Roles: Preliminary Description and Outline**.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Computer role-based access control models. *Computer*, *29*(2), 38–47. https://doi.org/10.1109/2.485845

Solanki, N., Zhu, W., Yen, I. L., Bastani, F., & Rezvani, E. (2016). Multi-tenant access and information flow control for SaaS. *Proceedings - 2016 IEEE International Conference on Web Services, ICWS 2016*, 99–106. https://doi.org/10.1109/ICWS.2016.21

Tang, B., Li, Q., & Sandhu, R. (2013). A multi-tenant RBAC model for collaborative cloud services. *2013 11th Annual Conference on Privacy, Security and Trust, PST 2013*, 229–238. https://doi.org/10.1109/PST.2013.6596058

Varadharajan, V., & Rai, S. (n.d.). *THIS WORK IS SUPPORTED BY COMMIT/ (A PUBLIC-PRIVATE RESEARCH COMMUNITY) PROGRAM. Policy Based Role Centric Attribute Based Access Control Model Policy RC-ABAC*.

Wang, G., Wang, Z., & Zhang, W. (2020). *A SaaS resource authorization management model based on resource directed tree*. 1747–1752. https://doi.org/10.1109/ITAIC49862.2020.9338841

Zhang, N., Ryan, M., & Guelev, D. P. (n.d.). *Evaluating Access Control Policies Through Model Checking*.

Zuo, Q., Xie, M., Qi, G., & Zhu, H. (2017a). Tenant-based access control model for multi-tenancy and sub-tenancy architecture in Software-as-a-Service. *Frontiers of Computer Science*, *11*(3), 465–484. https://doi.org/10.1007/S11704-016-5081-X

Zuo, Q., Xie, M., Qi, G., & Zhu, H. (2017b). Tenant-based access control model for multi-tenancy and sub-tenancy architecture in Software-as-a-Service. *Frontiers of Computer Science 2016 11:3*, *11*(3), 465–484. https://doi.org/10.1007/S11704-016-5081-X