ACCESS CONTROL IN MULTITENANT CLOUD ARCHITECTURE

1. Multi-tenant Cloud Computing

1.1 Introduction

An evolution in Internet technology, cloud computing is an advancement providing users with the means to access a wide range of computing power, software and platform as a service, as well infrastructure anytime, anywhere. Cloud computing enables on-demand network access to a shared pool of configurable computing resources, including servers, storage applications and services. Cloud computing capabilities have enabled businesses to offer services that seem to be infinitely scalable and elastic. Subscribers of cloud services can keep their upfront costs low by adjusting their level of service based on demand. It allows companies to start small and increase incrementally with demand. Although enterprises are attracted by the advantages of cloud services such as considerable storage resources and powerful computing ability. But cloud computing has a lot of problems need to overcome, such like the storage of Big Data, the traditional database structure has been unable to satisfy the huge amount of data queries. And the main topic of information security issues is a primary concern of IT managers. Because cloud computing providers cannot distribute servers to all enterprises, providers virtualize server clusters to distribute and rent computing resources to companies. This is named multi-tenancy technology.

Cloud computing is classified into private, public, hybrid, and community deployment models. These deployment model classifications are based on the infrastructure's ownership, management, and operation. Similarly cloud computing is also classified by service models namely: Software as a Service (**SaaS**), Platform as a Service (**PaaS**) and Infrastructure as a Service (**IaaS**), and each service model has different management and control that makes each unique. The SaaS delivers applications that are accessible to different users online, although, the user does not manage or control the underlying cloud infrastructure. While in the PaaS, the user has access and controls their data, the application, and the application development lifecycle, without control over the infrastructure. Whereas in the IaaS the user owns and manages the applications, data, operating system and application runtime. In this report we are focusing on SaaS applications on public cloud.

1.2 The key characteristics of a multi-tenant cloud applications

- **Sharing hardware resources**

Multi-tenancy is a computing architectural concept concerned with information sharing among multiple users referred to as tenants. In case cloud computing, improvement of resource utilisation and service availability in cloud computing are based on multitenancy. Multitenancy is a concept that enables sharing the same service instance, scaling up and down the resources allocated among different tenants. Both characteristics improve resource utilisation, cost and service availability. In addition, apart from the capacity of multi-tenancy to share resources as a strategy in cloud computing, it also enables service providers to maximise resource utilisation and, thus, reduce the servicing costs per tenant. In the database perspective, multi-tenancy as a principle where a single instance of the DBMS runs on a server, serving multiple clients (tenants). The multi-tenancy in database systems supports several separate and distinct groups of users, the users are referred to as tenants.

- **Requires high degree of configurability.**

Unlike single-tenant SaaS, multi-tenant setups cannot be completely customized since they are shared among different ten-ants. Therefore, it is necessary for multi-tenant software to be highly configurable. It needs to accommodate each client's settings, workflows among other. In a typical single-tenant setup, updates are usually made by creating branches in the development tree, but this does not work in multi-tenant setups and configuration needs to be part of the product design itself. SaaS like Email and CRM has the same workflow irrespective of which company is using the service. The service is standard across the board. However, for most SaaS, the workflow changes from client to client (tenant). As mentioned earlier, a key challenge in a multi-tenant environment is to provide a configurable setup for each tenant.

- **Databases**

SaaS applications typically run in the browser of a client (tenant) and can be accessed from anywhere. SaaS applications can range from a simple web mail application to a more complex CRM application. Since multi-tenant SaaS technology runs multiple service or application with same logical environment, it must ensure user's data isolation by application or service. There are three data isolation approaches applied to the cloud.

1. Separate Database: which is the simplest data isolation approach that stores each tenant data in a separate database.
2. Shared Database-Separate Schema: which hosts all the tenants in the same database instance, but each tenant has his own database schema.
3. Shared Database-Shared Schema: which allows tenants to store their data in the same database and same schema. In other words, a given table can store different table rows for different tenants, and a tenant ID column will differentiate and isolate the tenant's data.

These multi-tenant data isolation approaches have challenges in supporting highly manageable database schema, and in providing configurable database fields. These challenges are:

1. Isolating tenants' data by ensuring that each tenant can access only his own data.
2. Ensuring that the tenants' data is robust and secure.
3. Optimizing database performance.
4. Designing a database structure which works with different business domain applications.
5. Fulfilling different tenants' business requirements by using a tenant-aware data management based on Shared Database-Shared Schema approach.

In this report, we are focusing on the Shared Database-Shared Schema isolation approach, which requires a high degree of data isolation and configuration to ensure the security and privacy of tenants' shared data.

1.3 Challenges in Multi-tenancy

Security: One of the biggest challenges in multi-tenant cloud computing is ensuring the security and privacy of customer data because of the lack of filtration of the inside part of the servers because both the client and the attackers are on the same server. The shared nature of the infrastructure also means that a breach in one customer's data could potentially affect other customers. To address this challenge, cloud service providers must implement robust security measures, such as encryption, access controls, and monitoring.

Resource Allocation: In a multi-tenant environment, it is essential to ensure that resources are allocated fairly and efficiently. Customers should be able to access the resources they need without impacting the performance of other tenants. Cloud service providers must use sophisticated algorithms to manage resource allocation and ensure that customers receive the appropriate level of service.

Compliance: Multi-tenant cloud computing environments must comply with a range of regulatory and compliance requirements, such as data protection laws, industry standards, and security certifications. Cloud service providers must ensure that their infrastructure meets these requirements, and customers must be able to audit the provider's compliance efforts.

Network Congestion: In a multi-tenant environment, there is a risk of network congestion if multiple customers are accessing resources simultaneously. Cloud service providers must ensure that their infrastructure can handle the demands of multiple tenants without causing performance issues.

Data Isolation: Customers in a multi-tenant cloud computing environment must be able to trust that their data is isolated from other tenants. Cloud service providers must use advanced virtualization techniques to ensure that customer data is isolated from other tenants and that there is no risk of data leakage.

2. Access Control in Multi-tenant SaaS Applications

The main purpose of multi-tenancy is to separate user data and ensure that it is not influenced by other users. Thus, access control is critical in multi-tenant cloud computing because it allows cloud service providers to ensure that each customer has access only to the resources and data they are authorized to access. Multi-tenant cloud environments host multiple customers on shared infrastructure, and without proper access controls, a customer could potentially access another customer's data or resources, leading to privacy and security breaches.

Access control enables cloud service providers to limit access to resources based on a customer's identity and role within the organization. For example, a cloud service provider might limit a customer's access to specific data or applications based on the customer's job function, such as a developer or an administrator. Access control can be enforced using a variety of mechanisms in layered approach as follows:

Authentication verifies the identity of a user or system attempting to access a resource. In multi-tenant cloud environments, authentication is typically implemented using username and password combinations or other authentication methods such as multifactor authentication.

Authorization determines what resources a user or system can access based on their identity and the permissions assigned to their role. Authorization is often implemented using access control lists (ACLs) or role-based access control (RBAC) mechanisms.

Encryption can also be used to protect data in transit and at rest in multi-tenant cloud environments. By encrypting data, cloud service providers can ensure that only authorized users or systems can access the data, even if it is intercepted or stolen.

How does Access Control help resolve challenges in multi-tenant environment discussed before?

1. Security: Access controls can help ensure that only authorized users and systems have access to resources and data, protecting against unauthorized access and potential privacy and security breaches.
2. Compliance: Access controls can be used to enforce compliance with regulatory and compliance requirements by limiting access to resources and data based on the customer's identity and role within the organization.
3. Data Isolation: Access controls can help ensure that customer data is isolated from other tenants by limiting access to resources and data based on the customer's identity and role within the organization.
4. Network Congestion: Access controls can be used to manage resource allocation and ensure that customers receive the appropriate level of service, minimizing the risk of network congestion and performance issues.

3 Existing Approaches of Access Control

3.1 Role-Based Access Control (RBAC)

RBAC is a widely used access control model that is based on the principle of assigning roles to users and granting permissions to those roles. In multi-tenant cloud architectures, RBAC can be used to ensure that tenants only have access to the resources that they are authorized to access based on their assigned roles.

Most applications require multiple role definitions for different user functions. A role definition refers to a minimal set of privileges that users or programmatic components need in order to do their job. For example, business users and data analysts would typically have different set of permissions to allow minimum necessary access to resources that they use. This method has few advantages, such like users only have to adjust their roles if their role changed, system manager can reduce their job by the roles reduces the complex relation between users and objects. Thus, the process is simplified and costs are relatively low.

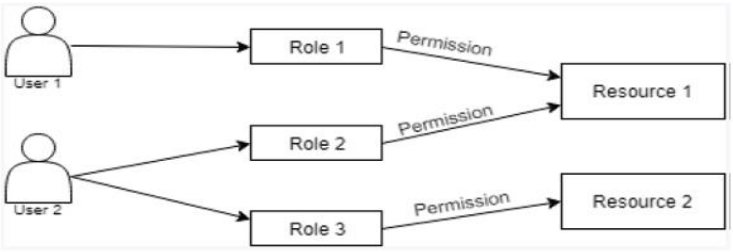


Figure 1 RBAC Structure.

In software-as-a-service (SaaS) applications, in addition to functional boundaries, there are also boundaries between tenant resources. As a result, the entire set of role definitions exists for each individual tenant. In highly dynamic environments (e.g., collaboration scenarios with cross-tenant access), new role definitions can be added ad-hoc. In such a case, the number of role definitions and their complexity can grow significantly as the system evolves. RBAC works well when you have a small number of tenants and relatively static policies.

Following are the limitations of RBAC Model:

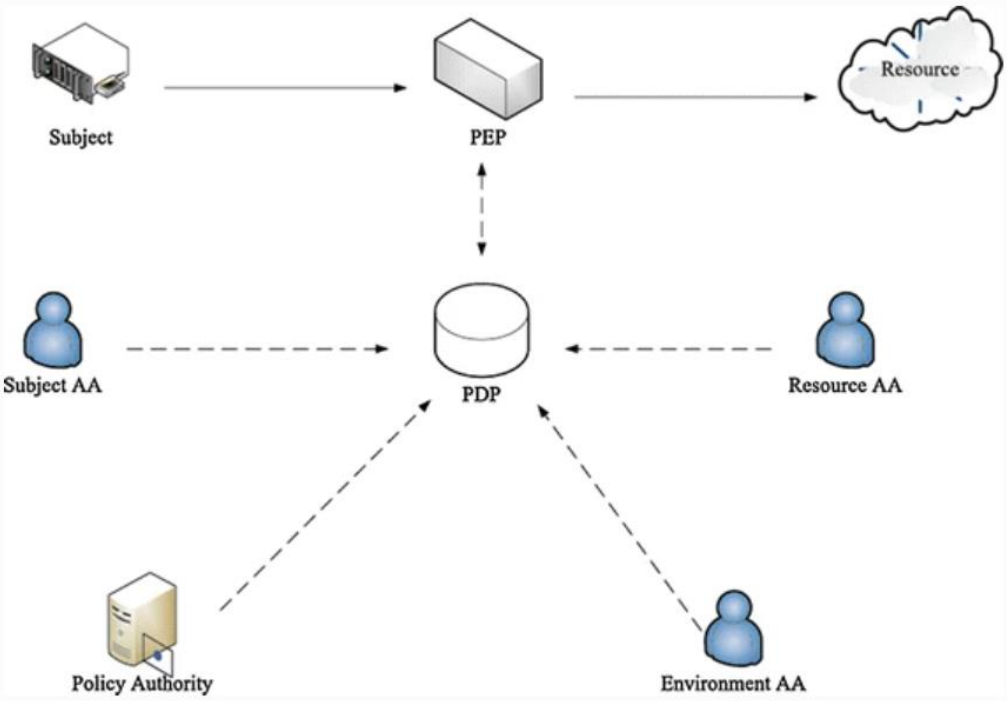
1. Limited granularity: RBAC assigns access based on predefined roles, which may not always provide the level of granularity needed in a multitenant environment. This can result in some users having access to resources they don't need, while others may not have enough access to perform their tasks.
2. Difficulty in managing large number of roles: In a multitenant environment, there can be a large number of roles to manage, which can be challenging to maintain and update. This can lead to confusion and inconsistencies in access control policies.
3. Limited flexibility: RBAC policies are often rigid and inflexible, which can make it difficult to adapt to changing business needs and security requirements.
4. Risk of role explosion: In a multitenant environment, the number of roles can quickly grow to a point where managing them becomes unfeasible. This is known as role explosion and can result in an overly complex and unmanageable access control system.
5. Limited support for dynamic environments: RBAC policies are typically static and do not easily accommodate changes in the dynamic nature of a multitenant environment. This can make it difficult to adapt to new tenants, applications, and services as they are added or removed from the system.

3.2 Attribute-Based Access Control (ABAC)

With the cloud service developing, the traditional RBAC mechanism can’t satisfy the large scale distribute system. ABAC is an access control model that is based on the principle of using attributes to determine access. In multi-tenant cloud architectures, ABAC can be used to enforce access control policies based on attributes such as tenant ID, user role, and resource type. This method is suitable for a wide range of SaaS applications, unless your use case requires support for frequently changed or added role definitions.

The ABAC mechanism is organized by two parts, the decision model and the architecture model. The attributes in these models are determined by the experts and they are dynamically altered constantly. In this manner, complying with the changing access control decisions become easier. The architecture model includes four main components: Attribute Authorities, AA, Policy Enforcement Point, PEP, Policy Decision Point, PDP and Policy Authority, PA.

Fig. 2



The AA is responsible for create and manage subject, resource and environment attributes. The PEP is responsible for request authorization decision and execution, PEP is widespread in the system for avoid conflict and ensure system is safe. The PDP is responsible for assess policy feasibility. The Pa is responsible for create and control access control policy. The policy may include description rule condition and another resource access restriction.

The flexibility achieved using this model increases the occurrences of policy conflicts and makes the maintenance and administration of the policies difficult. After a time period, the developers decided that ABAC is not efficient in system control because the access permissions given according to the attributes of the users became insufficient and unproductive. As a result, the Attribute and Role Based Access Control (ARBAC) model was implemented as the combination of ABAC and RBAC. According to the ARBAC, the system consists of four important elements and the privacy policy operations are performed via these elements [13].

3.3 Discretionary Access Control (DAC)

DAC is an access control model that is based on the principle of allowing users to determine access to resources. In multi-tenant cloud architectures, DAC can be used to allow tenants to set their own access control policies for their resources. Although the most important feature of this system is the fact that it has a high level of security, it cannot distinguish between the subjects and object domains and has security leaks. This access control is based on the user or group control. While assigning permissions and limitations to users, instead of working on a single user, it chooses to define groups that are controlled by the owner and introduces certain permissions and limitations to them. Of course, this situation can cause serious security leaks in the case that the owner is not trustworthy. The most important feature that can cause problems with the system is that the owner can transfer his authority to someone else.

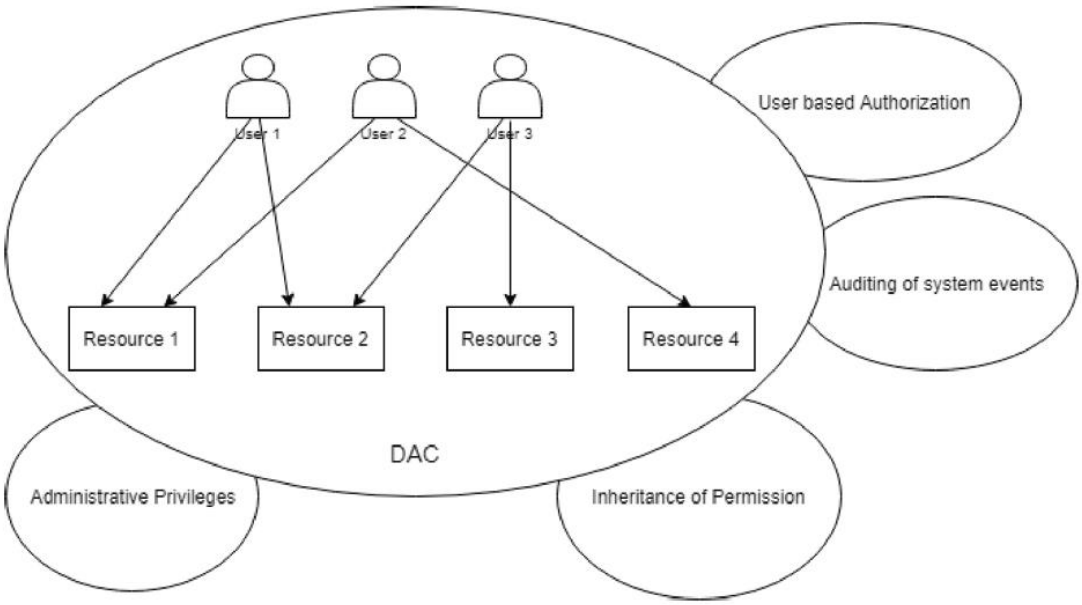


Figure 3 DAC Structure.

3.4 Mandatory Access Control (MAC)

MAC is an access control model that is based on the principle of assigning security labels to resources and users. In multi-tenant cloud architectures, MAC can be used to ensure that tenants can only access resources that have the appropriate security labels.

The decisions in MAC are not made by an owner but a central system. In this way, more powerful processes are performed in security models. The wholeness and privacy of the system are the most important features after security. Thus, it can increase the security to the highest level in a whole system. Its flexibility is really low. While MAC performs operations, it does not consider the relationship it has with the users into consideration, and for the security grant that ensures that all the users in the operating system perform the user assignments are considering this fact. It allows even the system administrative to perform operations according to the introduced policies by implementing limitations. The most important feature that values this system above DAC is the fact that MAC distinguishes between the subject and object.

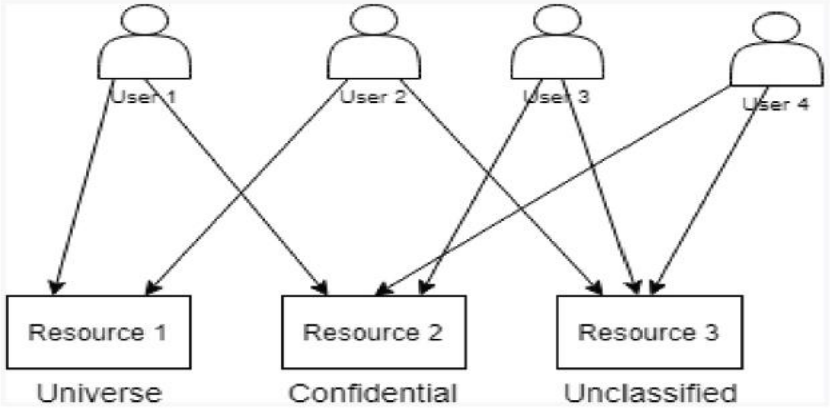


Figure 4 MAC Structure.

Along with above mentioned models, scholars have also proposed many hybrid access control models like:

- Temporal Role based Access Control (TR-BAC)
- Task-Role-based Access Control (T-RBAC)
- Privacy-Aware Role-based Access Control (P-RBAC)
- Attribute and Role Based Access Control (ARBAC)
- Hierarchical Attribute Based Access Control (HABE)

4. Proposed Solution

The fundamental goal of cloud storage security is to ensure the safety of data stored in the clouds, including the integrity, confidentiality and availability of data. The data stored in the clouds usually contains a large number of sensitive privacy data, such as patient records, customer information of enterprise etc. Once these private data are leaked, it will cause irreparable loss of economic and other influences to the society, groups and individuals. How to protect the confidentiality of sensitive data and guarantee the legality of users' accessing to data have become the most concerned problem for the users. Access control has naturally become the key technique of data privacy protection in cloud storage environment.

Existing access control models including those discussed above and other hybrid models in multi-tenant cloud applications have some limitations. They are as follows:

1. Role-based access control (RBAC) models may not provide enough granularity: RBAC models are based on assigning roles to users and giving them permissions based on those roles. However, in multi-tenant cloud applications, some users may require more granular access control than what RBAC models can provide.
2. Lack of support for dynamic access control: Many existing access control models are static, meaning that permissions are assigned to users based on their roles or other predefined criteria. In multi-tenant cloud applications, access control needs to be more dynamic, allowing for changes in access permissions as user roles or access requirements change.
3. Inability to handle complex relationships between tenants: In multi-tenant cloud applications, tenants may have complex relationships, such as parent-child or peer-to-peer relationships. Existing access control models may not be able to handle such relationships, leading to access control issues.
4. Limited support for fine-grained access control: Some multi-tenant cloud applications require fine-grained access control, such as controlling access at the data attribute level. Existing access control models may not provide such granularity, leading to security issues.
5. Limited support for auditing and compliance: Many existing access control models may not provide sufficient support for auditing and compliance requirements, such as logging access attempts and changes to access control policies.

To address these issues, we propose the framework presented by Elisa Bertino in the paper “Administration policies in a multipolicy authorization system” [14] for more flexible and adaptable approach to access control in multi-tenant cloud applications. In fact, the challenges faced by authorization systems in a multi-tenant cloud environment are similar to those faced by multipolicy authorization systems, such as conflicts between policies, inconsistency of policy administration, and the need for efficient policy evaluation.

The paper provides a policy administration framework that supports a large variety of administration policies, including centralized administration, owner-based administration, and joint-based administration. This framework can help cloud service providers address the challenges of implementing multiple access control policies and provide a flexible access control mechanism that supports different policies for different objects. It also offers a variety of administration policies to grant and revoke authorizations and provides a separation of specification and implementation.

The proposed framework can be used to implement flexible access control administration model for multi-tenant cloud architectures by defining a hierarchy of policies that includes global policies for the overall cloud environment, tenant-specific policies for individual tenants, and service-specific policies for different services

within the cloud. The main idea of this framework is to classify the subjects and objects in traditional access control mechanisms into two granule levels, one is tenant granule level, which is managed or controlled by clous service providers to implement the compartmentalization of different customers; the other is application granule level, which is controlled by customers to control the access to their applications. To ensure effective policy administration and enforcement, the proposed framework also includes a mechanism for policy conflict resolution, which can be used to resolve conflicts between different policies and ensure consistency and compliance in the multi-tenant cloud environment.

4.1 Key Features

The proposed framework is specifically designed for multi-policy authorization systems, which are commonly used in multi-tenant cloud architectures to manage access control policies for multiple tenants. This is different from traditional access control models, such as RBAC and ABAC, which are designed primarily for single-policy authorization systems.

This framework supports the use of dynamic policy enforcement, where policies can be updated in real-time based on changes in the environment. This can be especially useful in multi-tenant cloud architectures, where resource usage and user access can change rapidly.

One of the key features of the framework that supports scalability is the use of a hierarchy of policies, with higher-level policies defining the general guidelines for policy administration and lower-level policies implementing the details of policy enforcement. This allows for consistent policy enforcement across the cloud environment while allowing for flexibility and customization at the tenant level.

Another feature that supports scalability is the use of tenant-specific policies that can be customized to address the specific requirements of each tenant. This allows for strong isolation between tenants and helps to ensure that each tenant's policies are properly enforced.

4.2 Functional Analysis

The hierarchical structure of policies using this framework can be defined as follows:

- **Global policies**: These policies apply to the overall cloud environment and are applicable to all tenants. They define the general guidelines for policy administration, such as the acceptable use of resources, data retention policies, and compliance regulations.
- **Tenant-specific policies**: These policies are specific to individual tenants and are used to enforce access control and other security policies. They can be customized to address the specific requirements of each tenant, such as data location, encryption, and backup policies.
- **Service-specific policies**: These policies apply to specific services or applications within the cloud environment and can be used to define access control policies, usage policies, and other security policies for each service.

The hierarchical structure of policies can be implemented using a combination of policy languages, such as XACML (eXtensible Access Control Markup Language) and JSON (JavaScript Object Notation), and policy enforcement mechanisms, such as policy decision points (PDPs), policy enforcement points (PEPs), and policy information points (PIPs).

Following are the factors that we need to consider while implementing this framework to ensure that each tenant's policies and data are kept separate from other tenants:

Multi-Tenant Policy Definition Language: The policy definition language needs to support multi-tenancy so that each tenant can define and manage their own policies. The language should allow for policies to be defined at different levels, such as system-level policies and tenant-level policies. Tenant-level policies can be designed to define access controls specific to each tenant's resources and data, which helps to achieve tenant isolation.

Multi-Tenant Policy Administration Point: The policy administration point should provide a centralized location for managing policies across all tenants. However, it should also allow for tenant-specific policies to be managed

separately to maintain tenant isolation. In addition, the policy administration point should be designed to handle the high scalability requirements of a multi-tenant cloud environment.

Multi-Tenant Policy Decision Point: The policy decision point needs to be able to make access control decisions based on the policies defined for each tenant. It should also be designed to handle the high volume of requests that may come from multiple tenants concurrently.

Auditing and Reporting: A multi-tenant cloud environment requires auditing and reporting capabilities to track access to resources and detect any security breaches. The framework should support auditing and reporting across all tenants, while also allowing for tenant-specific reporting.

4.3 Security Analysis

Eavesdropping attack: This type of attack involves intercepting and listening to network traffic in order to obtain sensitive information, such as login credentials or data being transmitted between systems. By implementing the access control framework, tenant-specific policies can be defined to restrict access to sensitive data and resources to authorized users within the same tenant, thereby limiting the potential impact of eavesdropping attacks.

Replay attack: This type of attack involves capturing and replaying network traffic in order to perform unauthorized actions or gain access to sensitive information. By implementing the access control framework, tenant-specific policies can be defined to ensure that only authorized users within the same tenant are able to perform certain actions, such as accessing or modifying data, which can help to prevent replay attacks.

Man-in-the-middle (MITM) attack: This type of attack involves intercepting network traffic and altering it in order to gain access to sensitive information or perform unauthorized actions. By implementing the access control framework, tenant-specific policies can be defined to restrict access to sensitive data and resources to authorized users within the same tenant, which can help to prevent MITM attacks.

Forgery attack: This type of attack involves creating or modifying data in order to perform unauthorized actions or gain access to sensitive information. By implementing the access control framework, tenant-specific policies can be defined to ensure that only authorized users within the same tenant are able to modify or access certain data, which can help to prevent forgery attacks.

Offline guessing attack: This type of attack involves attempting to guess a user's login credentials or encryption keys in order to gain access to sensitive information. By implementing the access control framework, tenant-specific policies can be defined to enforce strong password requirements and limit the number of login attempts, which can help to prevent offline guessing attacks.

It's important to note that while the proposed access control framework can help mitigate these types of attacks to a certain extent, additional security measures (like authentication and authorization) may be necessary to fully protect a multi-tenant cloud computing environment from all potential threats.

References

1. Aulbach, S., Grust, T., Jacobs, D., Kemper, A., & Rittinger, J. (2008, June). Multi-tenant databases for software as a service: schema-mapping techniques. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 1195-1206). ACM.
2. Bezemer, C. P., & Zaidman, A. (2010, September). Multi-tenant SaaS applications: maintenance dream or nightmare?. In Proceedings of the Joint ERCIM Workshop on Soft-ware Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) (pp. 88-92). ACM.
3. Gerges, S., Khattab, S., Hassan, H., & Omara, F. (2013). Scalable multi-tenant authorization in highly collaborative cloud applications. International Journal of Cloud Computing and Services Science, 2(2), 106–115.
4. Ferraiolo, D. F., Sandhu, R., Garila, S., & Kuhn, D. R. (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, 4(3), 224–274.
5. Tang, B., Li, Q., & Sandhu, R. (2013). A multi-tenant RBAC model for collaborative cloud services. In Eleventh annual conference on pirvacy and trust.
6. Mon, E. E., & Naing, T. T. (2011). The privacy-aware access control system using attribute and role based access control in private cloud. In IEEE international conference on broadband network and multimedia technology.
7. Yuan, E., & Tong, J. (2005). Attributed based access control (ABAC) for web services. In IEEE international conference on web services.
8. Joshi, J., Bertino, E., Latif, U., & Ghafoor, A. (2005). A generalized temporal role-based access control. In IEEE transactions on knowledge and data engineering.
9. X. -Y. Li, Y. Shi, Y. Guo and W. Ma, "Multi-Tenancy Based Access Control in Cloud," 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 2010, pp. 1-4, doi: 10.1109/CISE.2010.5677061.
10. Servos, D., and Osborn, S. L. (2017). Current research and open problems in attribute-based access control. ACM Computing Surveys (CSUR), 49(4), 65.
11. Thomas, R. K., and Sandhu, R. S. (1993). Discretionary access control in object-oriented databases: Issues and research directions. In Proc. 16th National Computer Security Conference (pp. 63–74).
12. Zou, D., Shi, L., and Jin, H. (2009). DVM-MAC: a mandatory access control system in distributed virtual computing environment. In 15th International Conference on Parallel and Distributed Systems (ICPADS), (pp. 556–563). IEEE.
13. Lo, N.W., Yang, T.C. & Guo, M.H. An Attribute-Role Based Access Control Mechanism for Multi-tenancy Cloud Environment. Wireless Pers Commun 84, 2119–2134 (2015). https://doi.org/10.1007/s11277-015-2515-y
14. Bertino E, Ferrari E. Administration policies in a multipolicy authorization system. In: Proceedings of 11th IFIP WG 11.3 Conference on Database Security; 1997. p. 341–55.