NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 7 Issue 2, Pg. No. 86-99, AIJCST-V7I2P107, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I2P107

Original Article

Role-Based vs. Attribute-Based Access Control in Multi-Tenant Full-Stack Applications

*** Kiran Kumar Pappula[1], Guru Pramod Rusum[2], Sunil Anasuri[3]**
[1,2,3]Independent Researcher, USA.

Abstract:

The access control plays a critical role in securing the digital systems, especially multi-tenant full-stack applications contains various organizations or users operating under the same infrastructure. This article examines Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) in the context of applications. RBAC provides a straightforward way of assigning permissions depends upon the user roles, it is simple and easy to understand, but ABAC takes into consideration various attributes of users, resources, and the environments, giving a fine-grained level of control. Nonetheless, using one or another of these models in multi-tenant systems presents trade-offs that incorporate scalability, flexibility, complexity of enforcement, and overhead administration. In this paper, both RBAC and ABAC are considered in different multi-tenant full-stack settings. Since RBAC lacks the flexibility of policy enforcement provided by ABAC, we suggest a hybrid model; we would take the hierarchical characteristic nature of RBAC and combine it with the granularity of the issue that is presented by the acronym, which is ABAC. The steps will involve the development of a working prototype system, policy schema definition, incorporation of an authentication layer and experimentation on real-world datasets in order to emulate multi-tenant settings. Scalability, response time, and policy evaluation time are the metrics used in evaluation. Administrative overhead is also used as an evaluation metric. The most important conclusions are that ABAC is more flexible in dynamic, attribute-rich conditions, whereas RBAC does offer good policy enforcement practices and easy integration. The weaknesses of the two are alleviated in the hybrid approach, and contextual attributes can result in dynamic role assignment. There are also case studies of multi-tenant access management at various organizations with the help of various models. In the paper, future work on machine learning-based policy suggestion engines and interoperability frameworks across cloud ecosystems will be highlighted, as well as several recommendations about how to address the need.

Keywords:

Role-Based Access Control (Rbac), Attribute-Based Access Control (Abac), Multi-Tenant Applications, Full-Stack Development, Security Policies.

Article History:

Received: 30.01.2025

Revised: 01.03.2025

Accepted: 14.03.2025

Published: 22.03.2025

1. Introduction

The system security is built on access control mechanisms. They control the access to and conditions of access to resources by whom. The best-known and widely applied ones include Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC). In the case of full-stack multi-tenant applications or applications within which different user groups, or organizations (tenants), may have different access control constraints, a single instance of software is used by multiple groups of users. Scaling and flexibility of access control become imperative in these systems. [1-4] RBAC is based on the user's roles (e.g., Admin, Manager, Viewer) for access. Although RBAC proves easier to manage and enforce, it is not adaptable to dynamic or massive situations. In contrast, ABAC would allow attributes of the user, resources, and the environment (e.g. time of access, location) to be used when making a decision; this is more expressive but also introduces complexity to the system.

1.1. Attribute-Based Access Control in Multi-Tenant Full-Stack Applications

ABAC (Attribute-Based Access Control) has become a fundamental model for securing modern, full-stack, multi-tenant applications. Compared to the traditional Role-Based Access Control (RBAC), ABAC enables dynamic and fine-grained access control decisions making based on various attributes associated with users, resource, environment and action. This flexibility is especially important in full stack apps that may have various tenants and their requirements and policies.

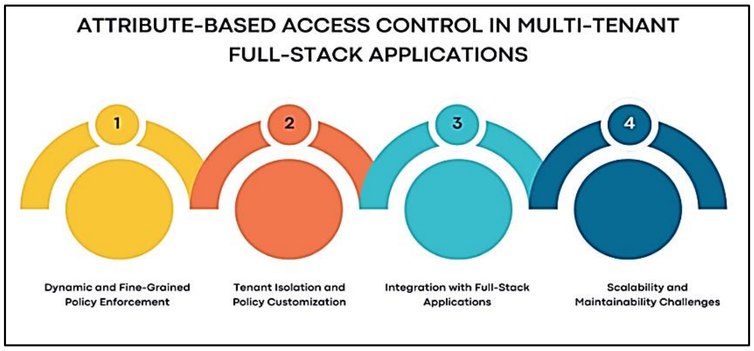
ATTRIBUTE-BASED ACCESS CONTROL IN MULTI-TENANT FULL-STACK APPLICATIONS
1
2
3
4
Dynamic and Fine-Grained Policy Enforcement
Tenant Isolation and Policy Customization
Integration with Full-Stack Applications
Scalability and Maintainability Challenges

Figure 1. Attribute-Based Access Control in Multi-Tenant Full-Stack Applications

1.1.1. Dynamic and Fine-Grained Policy Enforcement

In a multi-tenant architecture, tenants may require varied access levels for users, departments, or even resources. With ABAC, this degree of control can be achieved because attributes are evaluated at run-time. Illustratively, a policy may prevent users in the "finance" department from accessing only the expense reports during the so-called working hours. Such a dynamic assessment provides sensitivity to access decisions and tenant-oriented approaches.

1.1.2. Tenant Isolation and Policy Customization

Ensuring strict tenant isolation is one of the fundamental problems in multi-tenant systems, as users of one tenant cannot access the data (or other resources) of another. The ABAC facilitates tenant identifiers to policy rules as attributes. Every tenant specify the schemas and policies of attributes therefore customization is not restricted isolated or secured.

1.1.3. Integration with Full-Stack Applications

The ABAC is available through the stack in full-stack applications (in this case modern frontend frameworks and RESTful APIs). API endpoints are protected by the policies using backend middleware and policy engines. On the client side, UI components like button or section can be shown or hidden dynamically according to user's attributes (features) as it greatly improves User Experience and Security.

1.1.4. Scalability and Maintainability Challenges

Flexible though the ABAC might be, what it really highlights is the complexity of policy. The risks of misconfigurations and policy conflicts scales attributes and policies. The full-stack applications must integrate capabilities for policy checking, versioning, and debugging to make ABAC manageable.

1.2. Problem Statement

Even with advanced access control mechanisms, a few barriers exist to the efficient use of access control mechanisms in multi-tenant full-stack systems, especially in dynamic cloud-based environments. Traditional role-based access controls, such as Role-Based Access Control (RBAC), are simple and popular; however, they are not flexible enough to support context-aware situations and cannot adapt quickly to fast-changing access needs conversely, although more expressive and granular, Attribute-Based Access Control (ABAC) introduces significant policy complexity in terms of policy definition, evaluation, and maintenance, particularly as systems grow to support multiple tenants and services. [5,6] Such complexity usually leads to increased error tagging and low flexibility, and expectedly more time in terms of acquiring new tenants or administrators. In addition, the scalability of access control mechanisms using multi-tenant systems is a challenge. Every tenant is allowed to define its policies, roles, and access patterns, which can cause conflicts, redundancy, and impaired performance unless they are managed efficiently. With full-stack architectures where systems integrate front-end interfaces, back-end treatment, and APIs, it becomes increasingly difficult to ensure access control across all layers.

The overhead of incorporating integration that the alignment and synchronization of authentication tokens, policy logic, and session states can be high and prone to error, usually in environments which makes use of microservices or otherwise distributed components. There is also fundamental limitations of domain-specific constraints. Various application domains (e.g., healthcare, finance, education) can bear regulatory, organizational, or functional requirements that cannot be fulfilled very simply by general-purpose models. The present paper seeks to address these difficulties by identifying a hybrid access control model that is specifically tailored to accommodate multi-tenant full-stack systems. The model reflects both the scalability and ease of use of RBAC, as well as the flexibility of application based on the context of ABAC, with minimal integration and administration costs. We aim to evaluate this model within a simulated multi-tenant setting, which will attempt to demonstrate the resulting benefits of improved performance, flexibility, and usability, and form a basis for future research on scalable tenant-aware access control systems.

2. Literature Survey

2.1. Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is a popular access control system, the specifics of which were formally implemented in the early 2000s by the National Institute of Standards and Technology (NIST). [7-10] The basic idea behind RBAC is to assign specific responsibilities to users at an organizational level and hence grant the associated roles specific privileges rather than a particular individual user. The method makes it simpler to manage permissions, especially where an organization can have a lot of people, by generalizing access rights to manageable categories. RBAC can also dictate role hierarchies, where more privileged roles inherit the permissions granted to less privileged roles, and this scales well. The simplicity and ease of fitting the model into different enterprise systems contribute to its widespread adoption in diverse enterprise systems. Nevertheless, RBAC cannot be used in a dynamic or complex environment without restrictions. It is uncontext-aware and unable to adapt to real-time settings, e.g., the time of access and the user's location. Additionally, it has the role explosion problem, where a large number of roles must be created to handle all the combinations of permissions, resulting in administrative overhead.

2.2. Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC) offers a more flexible and dynamic alternative to RBAC, based on attributes associated with users, resources, and the controlled environment when making access decisions. These properties are applied to policies, and policies are characteristically expressed as Boolean expressions. Fine-grained control through ABAC is possible because it supports access rules that are quite specific, e.g., access is only granted during business hours or the user must have the same level or clearance as the requesting party. The model facilitates the dynamic enforcement of policies, which is why it is applicable to current, distributed systems that require context-aware decisions. ABAC, despite its benefits, introduces considerable complexity to the creation and management of policy. The preview arrays have the flexibility property that can result in very complex policies that cannot be built, tested, and debugged. In addition, implementation may be said to be challenging, whether in legacy systems or large-scale systems, due to the absence of standard policy languages and tools. The ABAC model is typically depicted as a mathematical function that takes into account the values of user characteristics, resource characteristics, and environmental factors, and based on these factors, it decides whether to grant access to a resource or not.

2.3. Access Control in Multi-Tenant Systems

Multi-tenant systems: Access control in a multi-tenant system has additional requirements that extend beyond the capabilities of existing models, such as RBAC and ABAC. Multi-tenancy allows many tenants (i.e., customers or organizational units) to share one common infrastructure and to have full data isolation and privacy. Tenant isolation is one of the primary challenges, as access control systems must ensure that a user of one tenant cannot access the data or resources of another. This is important to maintaining the confidentiality and complying with various laws, regulations etc. And there are also concerns about tenants' policies conflicting with each other. You need to be separated from these as each tenant can have it's own security policies, roles and even access policies. Struggles like that, may hinder well functioning of policy and lead to vulnerabilities we could have avoided. Leakage of data is also a major concern and a crucial mechanism for enforcing an effective definition of policies that require data to be segregated. To overcome these issues, multi-tenant systems often require tailor-made or mixed-model forms of access control, which incorporate aspects of RBAC, ABAC, and context-aware access control.

3. Methodology

3.1. System Architecture

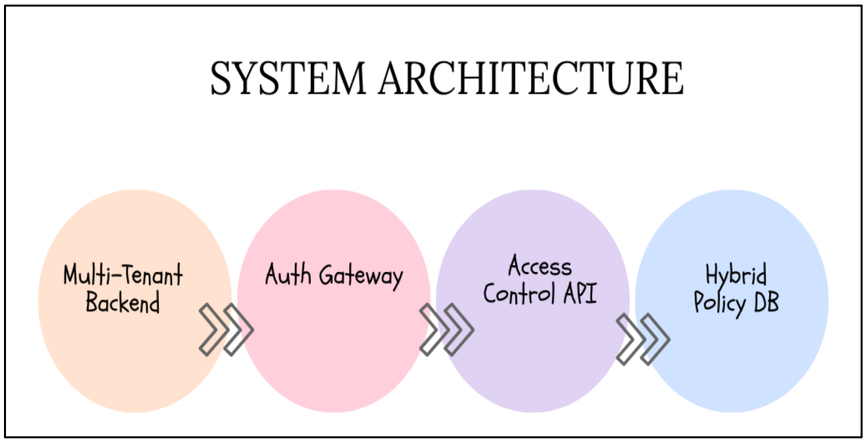

Figure 2. System Architecture

The system architecture specified in this paper aims to provide secure, isolated, and policy-driven access to resources in a multi-tenant environment. [11-14] It includes several important parts, each having specific duties that collaborate to bring inconsiderable and changing access determinations.

3.1.1. Multi-Tenant Backend

The Multi-Tenant Backend is the central point of the application infrastructure, storing data and services for numerous tenants in a shared space. The data and logic used by each tenant are also logically isolated, which means that no cross-tenant access is allowed. This is a middle layer that collaborates closely with the access control system to ensure that requests to access an item are checked, and the system operates on the item being guarded.

3.1.2. Auth Gateway

The Auth Gateway forms the interface to all authentication and authorization requests by the user. It clarifies the user's credentials, generates tokens (i.e., JWTs), and establishes session contexts. They also include the required metadata of identity and attributes on each request via the gateway to the access control API, which then performs the evaluation. It is easier to implement tenant-aware security when enforcement is concentrated.

3.1.3. Access Control API

The Access Control API mediates access to the residual URLs based on predetermined policies. It retrieves user and resource attributes from the Auth Gateway and verifies them against policies stored in the Hybrid Policy DB. This component serves as a policy decision point (PDP) and provides access decisions (permit or deny) to the upstream services. It has a modular structure that facilitates its interconnections with other modes of access controls, such as RBAC, ABAC, or even a hybrid model.

3.1.4. Hybrid Policy DB

The Hybrid Policy Database should maintain access control policies that incorporate both RBAC and ABAC elements. It enforces policy rules, role definitions, and attribute schemes that are tenant-specific. The design can facilitate dynamic access control, as it supports both role-based inheritance and attribute-based decision logic. The database is sliced or logically segregated according to tenant, allowing for scalability and preventing suspicious policy conflicts or data leakage.

3.2. Design Goals

The architecture of the proposed access control system has three major objectives, namely: facilitate dynamic enforcement of the policy, make possible the deployment of tenant-specific rules and last but not least, be maintainable (with manageable complexity). To start with, it is necessary to provide dynamic policy enforcement support in the case of modern cloud-based systems, where access decisions need to be dynamic based on real-time situational context, such as the user's location, the time of access, the device, or risk score. The long-established static access control systems, including RBAC, are not sufficient in these cases. The architecture, therefore, ought to be able to integrate elastic mechanisms, e.g. attribute-based or hybrid policies, to permit user and environmental attributes to be known at runtime. This confirms the access control decisions are not solely based on defined roles, rather on changing operational circumstances. Secondly, the tenant-specific rules enabled as multi-tenant architecture allows each tenant differ of access requirements, organizational structures, and security policies.

It would not do it on a one-size-fits-all basis would either be too permissive or restrictive. This system needs to give individual tenants ability to specify their roles, attributes, and policy access being under a shared enforcement model. The logical isolation and policy layers that tailored in tenant configurations mean each tenant configuration will not interfere another, to eliminate the risks of potential policy conflicts or data leaks. Last but not least, the system should not complicated it is unscalable, unfriendly, and difficult to administer. With the increase in size and scope of access control policies, those with dynamic attributes, misconfigurations and administrative overhead are more likely. The architecture permits designs are simple to define policies, fast policy execution engines, and entail reusable module are easy to upgrade and expand. These design goals, the system strikes a balance between flexibility, security, and simplicity of system operations are essential to secure access control in dynamic and multi-tenant environments.

3.3. Technology Stack

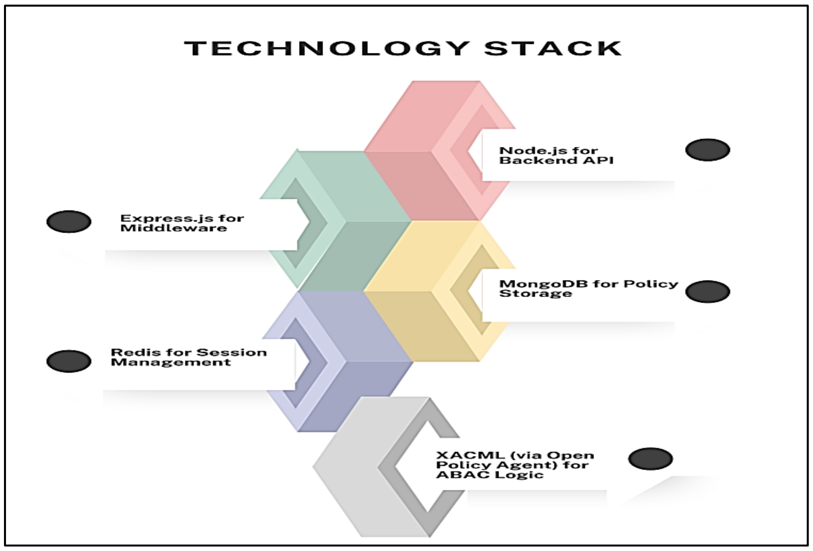
TECHNOLOGY STACK
Node.js for Backend API
Express.js for Middleware
MongoDB for Policy Storage
Redis for Session Management
XACML (via Open Policy Agent) for ABAC Logic

Figure 3. Technology Stack

The suggested solution will utilize a contemporary, scalable technology stack that is best suited for the development of secure, high-performance access control in a multi-tenant environment. All the parts are carefully chosen to serve a specific purpose in the architecture, enabling modularity, flexibility, and ease of incorporation.

3.3.1. Node.js for Backend API

The backend API utilizes Node.js as its runtime environment because it is an event-driven, non-blocking system that is well-suited for handling multiple concurrent requests in real-time. It is light weight yet powerful and with an extensive ecosystem for fast development and easy integration with other services. Node. js smoothly coordinates the interaction of other parts, for example, middleware making up the gateway to the authentication service, permission control logic or policy database.

3.3.2. Express.js for Middleware

The middleware framework top-level module of Express. js runs on the Node. js for structuring the back end app and dealing with HTTP routing. It makes it easy to process requests, perform token verification, validate input, and communicate with other services. Express has modularity, which enables it to be appropriate where security procedures should be integrated and tenant-aware request handling should be applied before relaying the requests to the access control API.

3.3.3. MongoDB for Policy Storage

MongoDB is selected as the primary repository for storing access control policies, user roles, and metadata of attributes. Due to its document-based, schema-less nature, it is flexible in representing dynamic and tenant-specific policy representations. The horizontal scale and high availability offered by MongoDB guarantee the responsiveness and reliability of the system, even under high loads of policy evaluations.

3.3.4. Redis for Session Management

Redis is used to manage sessions and cache tokens, providing in-memory storage to cache access tokens and user session data. Its low-latency work is essential for quickly validating sessions when performing checks on access control. Other features also provided by Redis, which can be used in session invalidation and synchronization of distributed components, include key expiration and pub/sub messaging.

3.3.5. XACML (for ABAC Logic)

The logic of attribute-based access control (ABAC) is based on XACML, which is implemented via the Open Policy Agent (OPA). OPA allows defining tricky policies on declarative languages such as Rego, which decides on user, resource, and environment attributes and generates the authorization decision. The inclusion of OPA in the system ensures standards-based, flexible, and auditable enforcement of a policy, allowing for dynamic and piecewise control of the policy over tenants.

3.4. Policy Schema

The proposed access control model has a policy schema compatible with the structure needed by Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC). This makes the system hybrid and it is suitable for the handling of flexible permissions in a multitenancy environment. [15–18] Access control is determined by the user's role in the RBAC model. The user is then given or denied access based on the permissions of that role. For instance, a policy like { "role": "admin", "permissions": ["read", "write", "delete"] } shows that a tenant with the admin role can be given full access, which includes the ability to read, write, and delete. This schema is simple and easy to use, and it works well for companies with strict hierarchies. You can use roles when you onboard new users, and you can change roles in one place without having to change each user's profile. This model works with operations that are either static or have well-defined roles. Conversely, ABAC has the advantages of being more dynamic and context-sensitive than a policy, as it examines the attributes of the user (subject), the resource sought by the user, environmental factors, and the action requested. An instance of ABAC policy can be as follows:

```
{
  "subject": { "department": "finance" },
  "resource": { "type": "report" },
  "condition": { "time": "working_hours" },
  "action": "read"
}
```

This policy users in the finance department browse a report-type resources during working hours. Such an attribute-based structure makes it possible to implement fine-grained control where administrators can attach rules that appreciate the real-world access constraints to an organization, like policies, compliance needs or risk levels. ABAC policies can be calculated at run time, which means that they can be used to support decisions that focus on real-time aspects, such as location, device trust, or user behavior. After including both RBAC and ABAC schemas in the system, the architecture is going to be simple but flexible enough for tenants such as cyber-XP's interest on having insight into their data so that they can decide exactly what control will be implemented over their information depending on their ongoing requirements and security stance.

3.5. Evaluation Metrics

In order to measure the efficiency and effectiveness of the proposed access control system, various evaluation metrics are considered. These metrics give details of the responsiveness, accuracy, scalability and manageability of system in a multi-tenant environment.

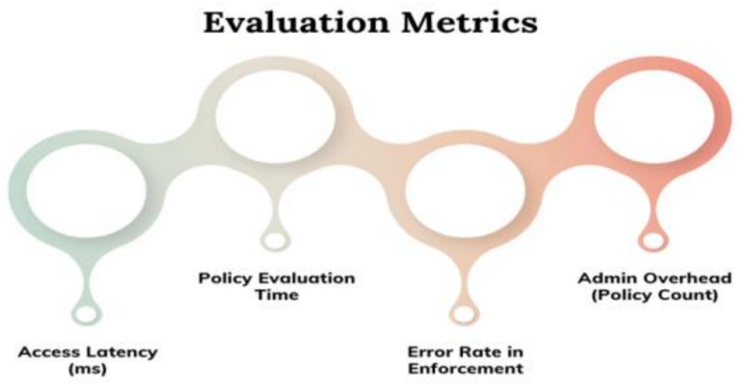

Figure 4. Evaluation Metrics

3.5.1. Access Latency (ms)

Access latency is calculated based on the sum total in milliseconds taken from the time a user submits a request until mement they receive either one of two responses ("yes" or "no"). This way is representative to dynamics of access control pipeline (i.e., authentication, policy look-up and decision). High-throughput and low-latency are essential for seamless user experience in traffic-intensive or real-time systems. The high access latency may occasionally indicate that the system design is "wearing" performance bottlenecks, e.g., slow DB queries or policy look-ups.

3.5.2. Policy Evaluation Time

The policy evaluation time look at access rules and decide. This is essential in systems with complex ABAC or mixed policies, whereas several attributes and conditions need to be evaluated on-the-fly. The efficient evaluation leads to timeliness of the answers and less system load. The trade-offs between the performance and policy hierarchies can also be revealed by the comparison of evaluation times for different types of policies (RBAC vs. ABAC).

3.5.3. Error Rate in Enforcement

The enforcement error is the percentage of decisions where the decision making entities incorrectly responds consist of the false positives (granting access to subjects who are not allowed) and false negatives (denying access to subjects who are allowed). An imprecise formulation of the policy, poorly defined role, or inaccurate attribute data all contribution to the high error rate. This measure is important to check that system used to access policies correctness and trust by group of users and administrates.

3.5.4. Error Rate in Enforcement

The number of total policies and roles to manage is a way of complexity and the ease of maintaining a system. When there are too many policies in a systems are inconvenient to administer the system costly due to mistakes made or values wasted. This metric provides the insights into the level of burden and scalability imposed for the system, especially in environments where there are many wielders and dynamic policy requirements.

4. Case Study and Evaluation

4.1. Scenario Description

We will also describe a rich case study based on a real-life multi-tenant example for showing the efficacy and applicability of the presented access control model. The focused testbed consists of a modelled multi-tenant Customer Relationship Management (CRM) application which is the organization's usual system for storing client, sales and internal operation data. The simulation has 5 different tenants (1,000 users each tenant to cover various organisation structures/access control needs/administrative domains/security policies). As such diversity allows us to study the access control models with adequately comparable workloads, and compare performance, flexibility and management overhead. Tenant A is purely operating on the traditional RBAC model: you can give users preset roles (e.g., sales_rep, manager, admin), and things get authorized according to those roles. Such a configuration would allow us not only to objectively determine the complexity and its computational efficiency of RBAC in terms of operating, but also to reveal its robustness on dynamic or context-dependent context.

Instead, Tenant B have only one ABAC PEP where user factors (i.e. which department the user belongs to, what is his clearance level and location) and resource condition functions or time conditions make access decisions. This set-up has some features on which this configuration is strong for ABAC: how fine-grained control and flexible can be done with ABAC, and how complex it is, and if we want to make computational expensive. Tenant C is provisioned with the enhanced hybrid model that integrates structural clearness of RBAC and dynamism of ABAC. In this way, it's possible to evaluate how the ability of hybrid model finding the trade-off between granularity policy and manageability in complex one. Tenant D implements access enforcement using AWS Identity and Access Management (IAM), which is a standard cloud-native access paradigm. Yet Tenant E employs Azure RBAC, Microsoft's role-based access model in its cloud ecosystem. Below are a few of the "cloud native" tools you can use to compare what you do vs. industry. Comparing these five policies against access latency, policy evaluation time, error rate and administrative effort shed light on the practical benefits and trade-offs of each approach at scale in a multi-tenant environment.

4.2. Experiment Setup

The advanced testing machine is created to guarante both high test quality and test control of the access system. The environment is deployed in AWS EC2 instances, and can scale-out, while also being isolated - thus serving as a multi-tenant simulation environment. In our test case, each tenant is running in a logically partitioned environment to ensure the right level of data isolation and drive us closer to making use of the underlying infrastructure seem more like the cloud model. The solution's back-end is developed with a Node. js API, since the asynchronous model and potential for multiple user access are superior. The API acts as the central layer where all access control logic, policy evaluation and data retrieval operations are performed. To resemble the real workload and user patterns, a synthetic dataset is developed. The dataset contains 100,000 records of HR data, including the employee's name, department, role, location, clearance level, and types of documents. These records are the resources to be secured in the system. They are accessed with the help of API requests, simulating user actions such as reading reports, updating entries, or retrieving sensitive information. The data is balanced among all five simulated tenants, providing comparable access control performance when compared against each other based on the different models (RBAC, ABAC, hybrid, AWS IAM, and Azure RBAC).

Postman is applied to create and simulate API requests that replicate the general API access patterns of users under various roles and characteristics when testing and automating. It can script request control, headers, and tokens, as well as conditions in variables, to replicate a complex access situation. Fine-grain details accumulate in Prometheus, such as a sub-API response time, how long an evaluation takes, the rate at which errors occur or the size of in-memory data structures – we can quantify behavior of our system.

4.3. Results Overview

The access latency results Comparison of Reaction Flow with Different Access Control Structures 115 of the generated by modeling of comparison for two different access control schemes in simulated multi-tenant CRM system. The access latency is also a key performance metric, which captures the processing time for users to request an operation and obtain a response from the system. A lower latency is thought to be more ideal because it means you can create a snappier and real interactive user experience.

Table 1. Access Latency (ms)

Model	Avg. Latency
RBAC	18 ms
ABAC	38 ms
Hybrid	25 ms
AWS IAM	22 ms
Azure RBAC	20 ms

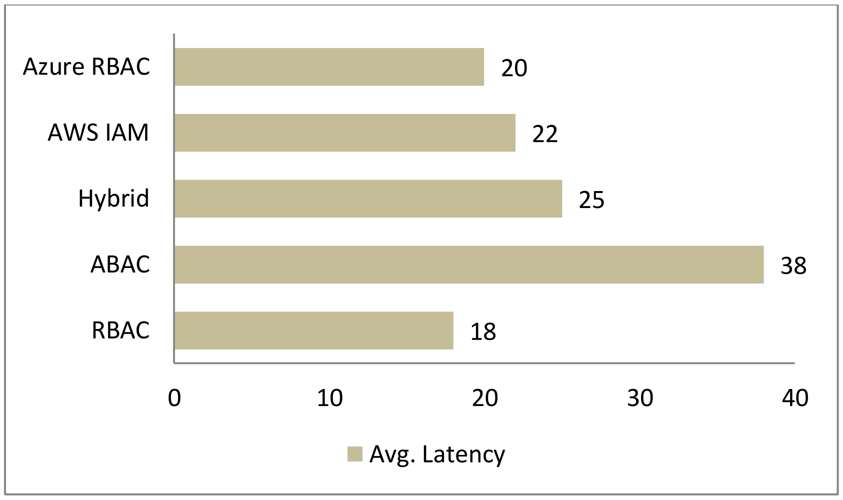

Figure 5. Graph Representing Access Latency (Ms)

4.3.1. RBAC – 18 ms

RBAC model had even lower average latency\footnoteOnce again a minimum average latency at 18ms shown. This can be assumed because of its simplicity when making decisions because only a predetermined set of permissions is checked with user roles. Dynamic context assessment is unsound, and this enables quick and consistent responses, which is why RBAC is suitable in an environment where changes in role and access requirements are modest.

4.3.2. ABAC – 38 ms

Conversely, the ABAC model had the maximum average latency of 38 milliseconds. This rise is explained by the fact that the assessment of various factors, i.e., user, resource, and environmental conditions, occurs on every request. The ABAC engine places a heavier emphasis on lookups and logical comparisons than the previous rules, which typically increases the overhead. However, this is a necessary trade-off to achieve the fine-grained and dynamic selectivity that the engine is designed to handle.

4.3.3. Hybrid – 25 ms

The hybrid model was a balanced-performing model, with an average latency of 25 milliseconds. It is designed to minimize the overhead that pure ABAC would have, and it offers the flexibility of contextual access control at the same time as relying on the structural efficiency of RBAC. This finding leads to the conclusion that the hybrid model may be an effective alternative to systems that not only need speed but also policy depth.

4.3.4. AWS IAM – 22 ms

The AWS IAM model had an average latency of 22 milliseconds and benefited from the cloud-provided intrinsic optimizations. IAM scalability, efficiencies IAM policies are designed to scale and be efficient, the AWS infrastructure that provides access evaluation for your resources has been optimised so even under complex scenarios of adding more related permissions, you will find the evaluation time is in milli seconds.

4.3.5. Azure RBAC – 20 ms

Last but not least, the average low latency of Azure RBAC was 20 ms, higher than RBAC and less than hybrid and ABAC. This only exemplifies good usage of role-based checks and, overall, an excellent Azure optimisation at cloud level, also adding that this platform is appropriate for medium-size dynamic requests at company scale.

5. Results and Discussion

5.1. Key Findings

The two important lessons have been learned from experimental cases that evaluate our multi-tenant access control system in various access control models. The results shows trade-offs among performance, flexibility, administration complexity offers insight for selecting or designing the access control mechanisms of cloud applications. The RBAC model presented this performance in response time, since it provided a stable performance and low latency. Despite its static character (with fixed responsibilities and predefined permissions), it meanwhile also supported quick decisions and a low computational overhead. But this boon is a bitter one leading to stiffness. RBAC is incapable of managing dynamic or context-sensitive access policies and thus is limited by more complex or time-sensitive security policies. Its strict constitution also makes it flexible to the variance in organizational requirements. Moreover, ABAC model grants much greater expressiveness and flexibility because many policies can have different types of attributes involved (role, department, resource type or environmental conditions such as time/date/location). This enables fine-grained, context-aware access control which is critical for dynamic and complex systems. However, the evaluation shows that ABAC introduces additional latency which is due to increased computational cost of calculating multiple attributes as well as evaluating Boolean policy logic. Furthermore debugging or administration of attribute-based policies might cause administrative overhead without control. The hybrid RBAC+ABAC model, which has been designed as a compromise among the models we discussed above is still perceived as a viable middle path, with acceptable trade-off between high performance and policy expressiveness. It had better performance than pure ABAC and more dynamic access than RBAC. This mix paradigm permits to have top-down role management structuring in case of this necessity, but also it authorizes for the situation require it, to go deeper on the attribute-based approach. That's why in multi-tenant scenarios wanting both low-work management and wide-ranged access policies, it becomes so cool!

5.2. Implications

The results of our study provide a few observations that can be used in the design and operation of access controls for multi-tenant environments. Most importantly, the results demonstrate that hybrid policy enforcement mechanisms (both RBAC- and ABAC-based) can be of a significant assistance for multi-tenant systems. This hybrid approach attempts to dampen these weaknesses of each model when used in isolation, by offering the administration friendliness of RBAC without having to choke down the necessary dynamism that ABAC brings to bear. This type of design is particularly suited to complex commonwealth spaces with a wide range of specific user needs that cannot be entirely captured by fixed schedules. The second observation that is worth mentioning here and we do so for the understanding of the reader is that sysadmins mostly use role based policies because they are transparent, easy-to-have and to maintain. Roles are very suitable against such organizational hierarchies, easily manageable in a very natural fashion especially when there is huge number of users to manage. However, sometimes roles are insufficient for any-one, even admin-sisteratorinoes; like time limited access, geographical location and device related requirements. When the above is not ideal, administrators require attribute based overrides to further restrict or extend accessibilities beyond role constraints. Thus, offering a hybrid model that enables both structured assignment to roles and checks of contextual attributes gives administrators the power to formulate more accurate and enforceable access policies without necessarily complicating the system substantially.

Furthermore, the experiments illustrate that policy versioning and rollback are essential in systems based on exercises with ABAC or hybrid models. Over the evolution of ABAC policies, they becomes complex and may increase the likelihood of misconfiguration where unintended users get access or service-impacting rules are implemented. Version controlled policy framework accomplishes this by allowing admins to view what has been changed and subsequently test policy in a stage environment before you deploy it or roll back if there are any issues. This feature is extensively improving the practical usability and stability of ABAC, making it more suited for deploying the system in a production setup.

5.3. Limitations

While the access control architecture is effective and practical according to the experimental results, there exist several limitations that should be addressed. The major limitation is that the prototype does not incorporate a production grade logging and audit system. In practice good logging is very important for monitoring access requests, troubleshooting policy misconfigurations, and compliance / auditing. Ordinary logging Since the current implementation of the prototype is limited to ordinary logging and weakly evaluates in isolation only, it barely responds to an enterprise setting which requires maximum traceability and accountability. Without logs that are detailed and explicit enough to give a chain of who accessed what, when and how under what circumstances, organizations have difficulty investigating incidents, enforcing security policies or demonstrating compliance with regulations such as GDPR, HIPAA or ISO 27001. The complexity of the ABAC rules is yet another key limitation. However, these rules provide significant power, which can easily become challenging to manage and validate at the scale of a system. ABAC policies are frequently written in such form that logical expressions are composed of several user, resource, and environmental characteristics. With hundreds of policies and policy attributes, depending on the company's size, it is more cumbersome to ensure that the policies do not conflict with each other, do not provide undesired access routes, and do not subject the result to security risks.

When it comes to ensuring policy correctness and consistency, they cannot be easily ensured with the help of formal verification when there are no formal mechanisms in place. Lacking the tools to automatically analyze, test, or verify the rules of ABAC, this can cause admins to unintentionally introduce bugs into the rules, resulting in over-permissive (or overly restrictive) access decisions. These constraints are the most essential areas to tackle the shift of the prototype into a production system. There are also options to improve in the future by incorporating policy versioning systems and advanced logging frameworks to give visibility and traceability. Also, adding formal techniques or policy tools, which include types of policy analyzers, static analyzers, or model checkers, can assist in verifying that the policies are valid before they are deployed, thus decreasing the likelihood of misconfiguration. Identification and resolution of such limitations will enhance the reliability, maintainability, and suitability of the respective system in an enterprise-ready multi-tenant environment.

6. Conclusion and Future Work

This paper then addresses the evolving requirements presented by access control in a multi-tenant environment, and works out the old and new policy models that stress with respect to performance, flexibility, and ease-ofmanagement[24]. We use architectural design, simulation, and empirical analysis to prove that neither of the two existing access control models – RBAC and ABAC – provides a complete solution to the realistic multi-tenancy requirements, which are dynamic in nature. Instead, the hybrid access control model that we are advocating will be effective in bleding the best of both approaches to lend roleoriented intuitiveness and contextual attribute-based awareness. In this paper, we aim to scale dynamic and customizable access control mechanisms which are high performing and easily tuned towards the sole needs of the tenant without imposing unacceptable levels of administrative overhead or performance deterioration.

The paper has the following few impactful contributions in access control. Firstly, it demonstrates the differences of RBAC and ABAC in a simulated and real-world multi-tenant CRM environment. This metaphor shows the sting of static role models, and the headache in working with heavy-attribute policies that fed into a more elegant solution. Second, our approach is model-driven: We assume a mixing policy model which mixes the hierarchical and fine-grained so that our hybrid model combining two advantages of RBAC and ABAC applications gives tenants access conditions more flexible than in both RBAC and ABAC applications but maintains transparency and scalability properties likewise. Finally, our goal is to validate the hybrid implementation in a controlled experiment which consider synthetic dataset and results shows that there are better usability and enforcement accuracy with no false alarms at acceptable latencies compared to low overhead administration.

There are still many ways the research could be taken further." One of the ways forward is to develop machine learning (ML) based methods enabling automatic generation of policy, and anomalous event detection. ML can suggest or generate policies by examining user behavior and access patterns, taking the administrative overhead off policy creation and enabling more adaptive policies. Another interesting direction is the research on decentralized identity solutions such as DIDs, which apparently could provide users with user-centric credentials and cross-platform processing of identity information. These approaches improve privacy and manoeuvrability in federated or multicloud systems. Finally, we are also going to extend our results towards cross-domain access control where users and services belong to multiple when they do not share a common trust domain. In such environments the requirement to enforce a consistent and secure policy is paramount, and we may use our hybrid solution to host inter-operable, federated access control systems.

References

[1] Sandhu, R. S. (1998). Role-based access control. In Advances in Computers (Vol. 46, pp. 237-286). Elsevier.

[2] Jin, X., Krishnan, R., & Sandhu, R. (2012, July). A unified attribute-based access control model covering DAC, MAC and RBAC. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 41-55). Berlin, Heidelberg: Springer Berlin Heidelberg.

[3] Yuan, E., & Tong, J. (2005, July). Attribute-based access control (ABAC) for web services. In the IEEE International Conference on Web Services (ICWS'05). IEEE.

[4] Hu, C. T. (2014). Attribute-Based Access Control (ABAC) Definition and Considerations.

[5] Servos, D., & Osborn, S. L. (2017). Current Research and Open Problems in Attribute-Based Access Control ACM Computing Surveys (CSUR), 49(4), 1-45.

[6] Zhang, Y., & Joshi, J. (2009). Access Control and Trust Management for Emerging Multidomain Environments (pp. 421-452). Emerald Group Publishing.

[7] Almutairi, A., Sarfraz, M., Basalamah, S., Aref, W., & Ghafoor, A. (2011). A distributed access control architecture for cloud computing. IEEE software, 29(2), 36-44.

[8] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications, 34(1), 1-11.

[9] Takabi, H., Joshi, J. B., & Ahn, G. J. (2010). Security and Privacy Challenges in Cloud Computing Environments. IEEE Security & Privacy, 8(6), 24-31.

[10] Sandhu, R., Robbins, K. A., White, G. B., Zhang, W., & Park, J. (2014). Multi-tenant access control for cloud services.

[11] Ulusoy, H., Colombo, P., Ferrari, E., Kantarcioglu, M., & Pattuk, E. (2015, April). GuardMR: Fine-grained security policy enforcement for MapReduce systems. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (pp. 285-296).

[12] Ahmad, W., Sunshine, J., Kaestner, C., & Wynne, A. (2015, October). Enforcing fine-grained security and privacy policies in an ecosystem within an ecosystem. In Proceedings of the 3rd International Workshop on Mobile Development Lifecycle (pp. 28-34).

[13] Indu, I., Anand, P. R., & Bhaskar, V. (2018). Identity and Access Management in Cloud Environments: Mechanisms and Challenges. Engineering science and technology, an international journal, 21(4), 574-588.

[14] El Sibai, R., Gemayel, N., Bou Abdo, J., & Demerjian, J. (2020). A survey on access control mechanisms for cloud computing. Transactions on Emerging Telecommunications Technologies, 31(2), e3720.

[15] Hayton, R. J., Bacon, J. M., & Moody, K. (1998, May). Access control in an open distributed environment. In Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186) (pp. 3-14). IEEE.

[16] Sandhu, R., Ferraiolo, D., & Kuhn, R. (2000, July). The NIST Model for Role-Based Access Control: Towards a Unified Standard. In ACM workshop on Role-based access control (Vol. 10, No. 344287.344301).

[17] Fatima, A., Ghazi, Y., Shibli, M. A., & Abassi, A. G. (2016). Towards Attribute-Centric Access Control: an ABAC versus RBAC argument. Security and Communication Networks, 9(16), 3152-3166.

[18] Lo, N. W., Yang, T. C., & Guo, M. H. (2015). An attribute-role-based access control mechanism for multi-tenancy cloud environments. Wireless Personal Communications, 84(3), 2119-2134.

[19] Khan, J. A. (2024). Role-based access control (RBAC) and attribute-based access control (ABAC). In Improving security, privacy, and trust in cloud computing (pp. 113-126). IGI Global Scientific Publishing.

[20] Solanki, N., Zhu, W., Yen, I. L., Bastani, F., & Rezvani, E. (2016, June). Multi-tenant access and information flow control for SaaS. In 2016, IEEE International Conference on Web Services (ICWS) (pp. 99-106). IEEE.

[21] Rusum, G. P., Pappula, K. K., & Anasuri, S. (2020). Constraint Solving at Scale: Optimizing Performance in Complex Parametric Assemblies. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(2), 47-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I2P106

[22] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105

[23] Enjam, G. R., & Tekale, K. M. (2020). Transitioning from Monolith to Microservices in Policy Administration. *International Journal of Emerging Research in Engineering and Technology*, *1*(3), 45-52. https://doi.org/10.63282/3050-922X.IJERETV1I3P106

[24] Pedda Muntala, P. S. R., & Jangam, S. K. (2021). Real-time Decision-Making in Fusion ERP Using Streaming Data and AI. *International Journal of Emerging Research in Engineering and Technology*, *2*(2), 55-63. https://doi.org/10.63282/3050-922X.IJERET-V2I2P108

[25] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107

[26] Enjam, G. R., & Chandragowda, S. C. (2021). RESTful API Design for Modular Insurance Platforms. *International Journal of Emerging Research in Engineering and Technology*, *2*(3), 71-78. https://doi.org/10.63282/3050-922X.IJERET-V2I3P108

[27] Karri, N., Pedda Muntala, P. S. R., & Jangam, S. K. (2021). Predictive Performance Tuning. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 67-76. https://doi.org/10.63282/3050-922X.IJERET-V2I1P108

[28] Rusum, G. P. (2022). Security-as-Code: Embedding Policy-Driven Security in CI/CD Workflows. *International Journal of AI, BigData, Computational and Management Studies*, *3*(2), 81-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I2P108

[29] Jangam, S. K., Karri, N., & Pedda Muntala, P. S. R. (2022). Advanced API Security Techniques and Service Management. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 63-74. https://doi.org/10.63282/3050-922X.IJERET-V3I4P108

[30] Anasuri, S. (2022). Zero-Trust Architectures for Multi-Cloud Environments. International Journal of Emerging Trends in Computer Science and Information Technology, 3(4), 64-76. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P107

[31] Pedda Muntala, P. S. R., & Karri, N. (2022). Using Oracle Fusion Analytics Warehouse (FAW) and ML to Improve KPI Visibility and Business Outcomes. International Journal of AI, BigData, Computational and Management Studies, *3*(1), 79-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P109

[32] Rahul, N. (2022). Optimizing Rating Engines through AI and Machine Learning: Revolutionizing Pricing Precision. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(3), 93-101. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I3P110

[33] Enjam, G. R. (2022). Secure Data Masking Strategies for Cloud-Native Insurance Systems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(2), 87-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I2P109

[34] Karri, N. (2022). AI-Powered Anomaly Detection. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(2), 122-131. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I2P114

[35] Tekale, K. M. T., & Enjam, G. reddy . (2022). The Evolving Landscape of Cyber Risk Coverage in P&C Policies. International Journal of Emerging Trends in Computer Science and Information Technology, 3(3), 117-126. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P113

[36] Rusum, G. P. (2023). Large Language Models in IDEs: Context-Aware Coding, Refactoring, and Documentation. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(2), 101-110. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P110
[37] Jangam, S. K. (2023). Importance of Encrypting Data in Transit and at Rest Using TLS and Other Security Protocols and API Security Best Practices. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 82-91. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P109
[38] Anasuri, S., & Pappula, K. K. (2023). Green HPC: Carbon-Aware Scheduling in Cloud Data Centers. *International Journal of Emerging Research in Engineering and Technology*, *4*(2), 106-114. https://doi.org/10.63282/3050-922X.IJERET-V4I2P111
[39] Reddy Pedda Muntala , P. S. (2023). Process Automation in Oracle Fusion Cloud Using AI Agents. International Journal of Emerging Research in Engineering and Technology, 4(4), 112-119. https://doi.org/10.63282/3050-922X.IJERET-V4I4P111
[40] Rahul, N. (2023). Personalizing Policies with AI: Improving Customer Experience and Risk Assessment. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 85-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P110
[41] Enjam, G. R. (2023). Optimizing PostgreSQL for High-Volume Insurance Transactions & Secure Backup and Restore Strategies for Databases. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 104-111. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P112
[42] Tekale, K. M., & Rahul, N. (2023). Blockchain and Smart Contracts in Claims Settlement. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(2), 121-130. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P112
[43] Karri, N. (2023). Intelligent Indexing Based on Usage Patterns and Query Frequency. International Journal of Emerging Trends in Computer Science and Information Technology, 4(2), 131-138. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P113
[44] Rusum, G. P., & Anasuri, S. (2024). AI-Augmented Cloud Cost Optimization: Automating FinOps with Predictive Intelligence. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(2), 82-94. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I2P110
[45] Enjam, G. R., & Tekale, K. M. (2024). Self-Healing Microservices for Insurance Platforms: A Fault-Tolerant Architecture Using AWS and PostgreSQL. International Journal of AI, BigData, Computational and Management Studies, 5(1), 127-136. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P113
[46] Rahul, N. (2024). Revolutionizing Medical Bill Reviews with AI: Enhancing Claims Processing Accuracy and Efficiency. International Journal of AI, BigData, Computational and Management Studies, 5(2), 128-140. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P113
[47] Partha Sarathi Reddy Pedda Muntala, "Enterprise AI Governance in Oracle ERP: Balancing Innovation with Risk" International Journal of Multidisciplinary on Science and Management, Vol. 1, No. 2, pp. 62-74, 2024.
[48] Jangam, S. K. (2024). Research on Firewalls, Intrusion Detection Systems, and Monitoring Solutions Compatible with QUIC's Encryption and Evolving Protocol Features . International Journal of AI, BigData, Computational and Management Studies, 5(2), 90-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P110
[49] Anasuri, S., Pappula, K. K., & Rusum, G. P. (2024). Sustainable Inventory Management Algorithms in SAP ERP Systems. International Journal of AI, BigData, Computational and Management Studies, 5(2), 117-127. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P112
[50] Karri, N. (2024). ML Algorithms that Dynamically Allocate CPU, Memory, and I/O Resources. *International Journal of AI, BigData, Computational and Management Studies*, *5*(1), 145-158. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P115
[51] Tekale, K. M., & Enjam, G. R. (2024). AI Liability Insurance: Covering Algorithmic Decision-Making Risks. International Journal of AI, BigData, Computational and Management Studies, 5(4), 151-159. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I4P116
[52] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106
[53] Enjam, G. R., & Chandragowda, S. C. (2020). Role-Based Access and Encryption in Multi-Tenant Insurance Architectures. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(4), 58-66. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P107
[54] Pedda Muntala, P. S. R. (2021). Integrating AI with Oracle Fusion ERP for Autonomous Financial Close. *International Journal of AI, BigData, Computational and Management Studies*, *2*(2), 76-86. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I2P109
[55] Rahul, N. (2021). Strengthening Fraud Prevention with AI in P&C Insurance: Enhancing Cyber Resilience. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 43-53. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P106
[56] Enjam, G. R., Chandragowda, S. C., & Tekale, K. M. (2021). Loss Ratio Optimization using Data-Driven Portfolio Segmentation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 54-62. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P107
[57] Karri, N. (2021). AI-Powered Query Optimization. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 2(1), 63-71. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P108
[58] Rusum, G. P., & Pappula, kiran K. . (2022). Event-Driven Architecture Patterns for Real-Time, Reactive Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(3), 108-116. https://doi.org/10.63282/3050-922X.IJERET-V3I3P111
[59] Jangam, S. K. (2022). Role of AI and ML in Enhancing Self-Healing Capabilities, Including Predictive Analysis and Automated Recovery. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 47-56. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P106
[60] Anasuri, S., Rusum, G. P., & Pappula, kiran K. (2022). Blockchain-Based Identity Management in Decentralized Applications. International Journal of AI, BigData, Computational and Management Studies, *3*(3), 70-81. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I3P109
[61] Pedda Muntala, P. S. R. (2022). Enhancing Financial Close with ML: Oracle Fusion Cloud Financials Case Study. *International Journal of AI, BigData, Computational and Management Studies*, *3*(3), 62-69. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I3P108

[62] Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
[63] Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110
[64] Karri, N., Jangam, S. K., & Pedda Muntala, P. S. R. (2022). Using ML Models to Detect Unusual Database Activity or Performance Degradation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(3), 102-110. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I3P111
[65] Tekale, K. M. (2022). Claims Optimization in a High-Inflation Environment Provide Frameworks for Leveraging Automation and Predictive Analytics to Reduce Claims Leakage and Accelerate Settlements. International Journal of Emerging Research in Engineering and Technology, 3(2), 110-122. https://doi.org/10.63282/3050-922X.IJERET-V3I2P112
[66] Rusum, G. P. (2023). Secure Software Supply Chains: Managing Dependencies in an AI-Augmented Dev World. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(3), 85-97. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I3P110
[67] Jangam, S. K., & Karri, N. (2023). Robust Error Handling, Logging, and Monitoring Mechanisms to Effectively Detect and Troubleshoot Integration Issues in MuleSoft and Salesforce Integrations. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 80-89. https://doi.org/10.63282/3050-922X.IJERET-V4I4P108
[68] Anasuri, S. (2023). Synthetic Identity Detection Using Graph Neural Networks. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 4(4), 87-96. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P110
[69] Reddy Pedda Muntala, P. S., & Karri, N. (2023). Voice-Enabled ERP: Integrating Oracle Digital Assistant with Fusion ERP for Hands-Free Operations. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(2), 111-120. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P111
[70] Rahul, N. (2023). Transforming Underwriting with AI: Evolving Risk Assessment and Policy Pricing in P&C Insurance. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 92-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P110
[71] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2023). Zero-Downtime CI/CD Production Deployments for Insurance SaaS Using Blue/Green Deployments. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 98-106. https://doi.org/10.63282/3050-922X.IJERET-V4I3P111
[72] Tekale, K. M. (2023). Cyber Insurance Evolution: Addressing Ransomware and Supply Chain Risks. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 124-133. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I3P113
[73] Karri, N., & Jangam, S. K. (2023). Role of AI in Database Security. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 4(1), 89-97. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I1P110
[74] Rusum, G. P. (2024). Trustworthy AI in Software Systems: From Explainability to Regulatory Compliance. International Journal of Emerging Research in Engineering and Technology, 5(1), 71-81. https://doi.org/10.63282/3050-922X.IJERET-V5I1P109
[75] Enjam, G. R. (2024). AI-Powered API Gateways for Adaptive Rate Limiting and Threat Detection. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(4), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I4P112
[76] Rahul, N. (2024). Improving Policy Integrity with AI: Detecting Fraud in Policy Issuance and Claims. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P111
[77] Reddy Pedda Muntala, P. S., & Karri, N. (2024). Autonomous Error Detection and Self-Healing Capabilities in Oracle Fusion Middleware. International Journal of Emerging Research in Engineering and Technology, 5(1), 60-70. https://doi.org/10.63282/3050-922X.IJERET-V5I1P108
[78] Karri, N., & Jangam, S. K. (2024). Semantic Search with AI Vector Search. International Journal of AI, BigData, Computational and Management Studies, 5(2), 141-150. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P114
[79] Tekale, K. M., & Rahul, N. (2024). AI Bias Mitigation in Insurance Pricing and Claims Decisions. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 138-148. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P113
[80] Jangam, S. K. (2024). Advancements and Challenges in Using AI and ML to Improve API Testing Efficiency, Coverage, and Effectiveness. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(2), 95-106. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I2P111
[81] Anasuri, S. (2024). Secure Software Development Life Cycle (SSDLC) for AI-Based Applications. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 104-116. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P110.