CognitivaTech: Ingeniería de Software Inteligente y Sistemas Adaptativos. 2024; 1(1):3
doi: 10.63688/cognitivatech.v1.i1.3

ORIGINAL

Methodological model for authorization management in microservices-based applications

Modelo metodológico para la gestión de autorización en aplicaciones basadas en microservicios

Daniel Oswaldo Ramírez Guevara[1]

[1] Instituto Sapiens de Investigación Científica Multidisciplinar. Milagro, Ecuador.

Submitted: 20-05-2024 | Revised: 03-06-2024 | Accepted: 20-07-2024 | Published: 22-07-2024

Corresponding Author: Daniel Oswaldo Ramírez Guevara

ABSTRACT

The growing adoption of microservices-based architectures has transformed software development by enabling distributed, scalable, and loosely coupled systems; however, this evolution has also increased the complexity of access control management, particularly in terms of authorization. In this context, this research proposes a methodological approach to systematically integrate authorization policies into the software development lifecycle of microservice-based applications. The study follows a descriptive–analytical design structured into four stages: authorization requirements analysis, policy formalization, implementation within a distributed architecture, and validation through a case study. The results show that the use of the Attribute-Based Access Control (ABAC) model allows the definition of more precise and flexible rules compared to traditional role-based approaches, reducing redundancy and improving system adaptability. Additionally, separating authorization logic from business logic proved essential for enhancing maintainability and scalability, enabling policy modifications without affecting the services. The validation phase demonstrated that the early integration of authorization helps maintain consistency across requirements, design, and implementation, avoiding inconsistencies and rework. However, the findings also indicate that properly defining attributes requires a rigorous initial analysis. In conclusion, the proposed approach improves authorization management in microservices by providing a structured, coherent, and adaptable framework, contributing to the development of more secure and efficient systems.

Keywords: microservices; access control; ABAC; software security.

RESUMEN

La creciente adopción de arquitecturas basadas en microservicios ha transformado el desarrollo de software al favorecer sistemas distribuidos, escalables y desacoplados; sin embargo, esta evolución ha incrementado la complejidad en la gestión del control de acceso, especialmente en lo referente a la autorización. En este contexto, la presente investigación propone un enfoque metodológico para integrar de manera sistemática políticas de autorización dentro del ciclo de desarrollo de aplicaciones basadas en microservicios. El estudio adopta un diseño descriptivo–analítico, estructurado en cuatro etapas: análisis de requisitos de autorización, formalización de políticas, implementación en una arquitectura distribuida y validación mediante un caso de estudio. Como resultado, se identificó que el uso del modelo basado en atributos (ABAC) permite definir reglas más precisas y flexibles en comparación con enfoques tradicionales basados en roles, reduciendo la redundancia y mejorando la adaptabilidad del sistema. Asimismo, la separación de la lógica de autorización respecto a la lógica de negocio demostró ser clave para mejorar la mantenibilidad y escalabilidad, facilitando la modificación de políticas sin afectar los servicios. La validación evidenció que la integración temprana de la autorización contribuye a mantener la coherencia entre requisitos, diseño e implementación, evitando inconsistencias y retrabajos. No obstante, se identificó que la correcta definición de atributos requiere un análisis inicial riguroso. En conclusión, el enfoque propuesto mejora la gestión de la autorización en microservicios al proporcionar un marco estructurado, coherente y adaptable, contribuyendo al desarrollo de sistemas más seguros y eficientes.

Palabras clave: microservicios; control de acceso; ABAC; seguridad de software.

INTRODUCTION

The adoption of microservices-based architectures has significantly transformed the development of modern software systems, enabling the replacement of monolithic structures with sets of independent, specialized, and highly scalable services (Newman, 2015; Sidler et al., 2022). This approach facilitates modularity, component reuse, and the continuous evolution of applications, as each microservice can be developed, deployed, and maintained autonomously. Furthermore, interactions between these services are conducted through well-defined interfaces, generally via APIs, which promotes interoperability and integration in distributed environments (Fielding, 2000).

Nevertheless, this decentralization of business logic introduces new challenges, particularly regarding security management. As the number of services increases and communication between them intensifies, the complexity of ensuring adequate access control also rises (Yarygina & Bagge, 2018). In this context, aspects such as authentication and authorization become fundamental, as they determine who can access system resources and under what conditions (Gollmann, 2010). The distributed nature of microservices complicates the implementation of centralized access control mechanisms, necessitating the adoption of more flexible and granular approaches (Chandramouli, 2019).

One of the main challenges lies in defining authorization policies that are sufficiently expressive to adapt to different contexts, yet structured and systematic enough to facilitate their implementation and maintenance. In particular, attribute-based access control models have emerged as a suitable alternative in dynamic environments, as they allow decisions to be made based on multiple characteristics of the user, resource, and environment (Hu et al., 2015; Yuan & Tong, 2005). However, their integration within the software development lifecycle still presents limitations, due to the lack of clear methodologies guiding their definition from the early design stages (Sänger & Abeck, 2023).

In many cases, security mechanisms are incorporated late in the development process, which can lead to inconsistencies, vulnerabilities, and higher implementation costs (Devanbu & Stubblebine, 2000). Therefore, it is necessary to address authorization as a non-functional requirement from the earliest phases of development, integrating it throughout the analysis, design, and implementation of the system (Firesmith, 2003). This involves not only defining what must be protected, but also establishing how access policies are formulated, implemented, and validated in distributed environments.

In this regard, the present approach proposes a comprehensive perspective for authorization management in microservices-based applications, aimed at structuring policy definition, facilitating technical implementation, and promoting systematic incorporation within the software development process. In this way, the goal is to reduce the inherent complexity of these systems, improve security, and ensure coherent management of resource access in modern architectures.

METHODS

This research adopts an applied approach with a descriptive–analytical design, aimed at structuring and evaluating a systematic process for the definition, implementation, and integration of authorization policies in microservices-based architectures. The methodology is constructed by aligning concepts from software engineering, information security, and access control, with an emphasis on attribute-based models (ABAC) and their incorporation within the development lifecycle.

In the first phase, a targeted review of technical and scientific literature is conducted, focusing on three areas: microservices architectures, access control mechanisms, and security strategies in distributed systems. This review enables the identification of limitations in the formulation of authorization policies and their integration into development processes, particularly in environments where decoupled services and API-based communication predominate.

Based on this analysis, a methodological model structured in four stages is defined, which addresses the need to integrate authorization as a non-functional requirement from the early phases of development:

Análisis de requisitos de autorización

The system actors, protected resources, and associated operations (for example, CRUD operations in REST services) are identified.

In this stage, relevant attributes of the subject, resource, and context are established, which serve as the basis for constructing ABAC policies. The specification is carried out in structured natural language, ensuring clarity and traceability with functional requirements.

Formalización de políticas de acceso

The defined requirements are transformed into formal policies through a rule-based syntactic structure. This formalization reduces ambiguities and facilitates subsequent technical implementation. An intermediate representation is used that organizes conditions, attributes, and access decisions, allowing translation into policy languages such as those used in authorization engines.

Implementación en arquitectura de microservicios

Validación mediante caso de estudio

The proposed model is evaluated through a representative case study, in which a microservices-based application with granular access control requirements is implemented. Aspects such as policy consistency, ease of integration, maintainability, and adaptability to changing requirements are analyzed. Validation focuses on verifying the feasibility of the approach and its contribution to reducing the complexity of authorization management.

Finally, the results were analyzed qualitatively, considering criteria such as coherence between requirements and policies, separation of responsibilities, and alignment with security principles in distributed systems. This methodological approach not only enables the structuring of authorization in microservices but also allows for its systematic integration within the development process, addressing one of the main limitations identified in the current state.

RESULTS

The application of the methodological model enabled a consistent structuring of the definition and implementation of authorization policies in a microservices-based architecture. The results are presented according to the proposed stages: specification, formalization, implementation, and validation.

Resultados del análisis de requisitos

Three main types of actors were identified (standard user, administrator, and internal service), as well as five critical resources (users, orders, reports, configuration, and audit). Based on these elements, relevant attributes for the ABAC model were defined, distinguishing between subject, resource, and environment attributes.

These attributes enabled the construction of more expressive rules compared to traditional models based solely on roles.

Resultados de la formalización de políticas

Table 1. Example of attributes defined for ABAC policies

Category	Attribute	Description
Subject	role	User type (admin, user, service)
Subject	user_id	Unique user identifier
Resource	resource_type	Entity type (order, report, etc.)
Resource	owner	Resource owner user
Environment	access_time	Time at which the request is made
Environment	source_ip	Source IP address

The authorization policies were transformed from natural language descriptions into formal structures based on conditional rules. This eliminated ambiguities and facilitated their implementation in policy engines.

Table 2. Example of policy transformation

Natural language	Formal policy (logical structure)
A user can view their own orders.	allow if (role = user) ∧ (user_id = owner)
An administrator can access all reports.	allow if (role = admin)
Access is restricted outside of working hours.	deny if (access_time < 08:00 ∨ access_time > 18:00)

The formalization demonstrated that the use of attributes allows for the definition of more precise conditions, reducing redundant rules.

Resultados de la implementación

The policies were implemented using an external authorization component decoupled from the microservices. This component evaluated incoming requests before granting access to the services.

The following observations were made:

- The separation between business logic and authorization reduced code complexity in the microservices.
- Policies could be modified without the need to redeploy the services.
- The validation of tokens and attributes enabled uniform enforcement of access control on each request.

Resultados de la validación

The case study demonstrated that the model enables coherent management of authorization policies in distributed systems. Scenarios involving multiple services interacting with each other were evaluated, where each request had to be validated according to the defined policies.

Table 3. Implementation evaluation

Criterion	Observed Result
Coupling	Low (authorization logic is decoupled)
Scalability	High (component is reusable)
Maintainability	High (policies are centralized)
Flexibility	High (adaptable to new attributes)

The following findings were identified:

- Coherence between requirements and policies was maintained throughout the entire development cycle.
- The early incorporation of authorization prevented inconsistencies in later phases.
- The use of ABAC allowed for the management of complex scenarios without significantly increasing the number of rules.

Table 4. Comparison before vs after the model

Aspect	Traditional Approach	Proposed Model
Policy Definition	Unstructured	Structured and traceable
Integration	Late	From the analysis phase
Access Control	Role-based	Attribute-based (ABAC)
Adaptability	Limited	High

Overall, the results indicate that the proposed model improves clarity in policy definition, facilitates their implementation in microservices environments, and reduces the complexity associated with access control management in distributed systems.

DISCUSSION

The results demonstrate that the structured incorporation of authorization policies within the microservices development cycle not only improves system organization but also reduces typical problems associated with security management in distributed environments. Unlike traditional approaches, where access control is implemented reactively, the applied model enabled the establishment of a direct relationship between initial requirements and the policies ultimately deployed.

One of the most relevant aspects is the effectiveness of the attribute-based access control (ABAC) model in representing real-world scenarios. The results show that the use of subject, resource, and environment attributes allowed for the definition of more precise rules without the need to significantly increase the number of policies. This contrasts with models based solely on roles, where scalability is often affected by the proliferation of permission combinations. In this regard, the reduction of redundancy observed in the formalization confirms that ABAC is better suited to the dynamic nature of microservices.

Additionally, the separation of authorization logic from business logic proved to be a key factor in system maintainability. The implementation results indicate that changes in policies did not require modifications to the services, which reduces the risk of introducing errors and facilitates system evolution. This finding is consistent with the decoupling principles characteristic of microservices architectures, where each component should be able to evolve independently.

However, the results also highlight certain challenges. The need to define appropriate attributes from the analysis phase implies a greater initial effort compared to simpler approaches. If attributes are not correctly identified, policies may become incomplete or excessively complex. Furthermore, although policy centralization improves management, it introduces a critical dependency on the authorization component, which requires ensuring its availability and performance.

Regarding validation, the case study confirmed that the early integration of authorization helps maintain system consistency throughout development. Traceability between requirements, policies, and their implementation enabled the early identification of inconsistencies, thereby preventing rework in later stages. These results suggest that treating authorization as a non-functional requirement from the outset not only enhances security but also improves the overall quality of the development process.

Overall, the discussion of the results indicates that the proposed model achieves a balance between flexibility and control in authorization management for microservices. Although it introduces some complexity in the initial phases, the benefits in terms of maintainability, scalability, and consistency justify its adoption in distributed systems that require granular access control.

CONCLUSIONS

This research demonstrated that authorization management in microservices-based architectures can be addressed more effectively when it is structurally integrated into the software development lifecycle. Through the proposed model, a clear relationship was established between authorization requirements, their formalization, and their implementation, reducing ambiguities and improving system consistency.

The results show that the use of an attribute-based approach (ABAC) provides greater adaptability to dynamic scenarios, allowing for the definition of more precise policies without unnecessarily increasing complexity. Similarly, the separation between business logic and authorization logic contributed significantly to the maintainability and scalability of the architecture, facilitating system evolution without affecting its main components.

The early integration of authorization as a non-functional requirement was consolidated as a key factor in avoiding inconsistencies and rework in later stages of development. This approach not only strengthens system security but also improves the quality of the software engineering process by promoting a more comprehensive and systematic perspective.

Nevertheless, it was identified that the correct definition of attributes and policies requires considerable initial effort, which implies the need for clear methodological guidelines and trained personnel. Despite this, the benefits obtained in terms of control, flexibility, and consistency justify the adoption of the model in distributed environments.

In conclusion, the proposed approach constitutes a viable alternative to address one of the main challenges in microservices architectures: the implementation of robust, scalable authorization mechanisms aligned with the development process, thereby contributing to the construction of more secure and sustainable systems.

REFERENCES

1. Chandramouli, R., Butcher, Z., & Chetal, A. (2021). Attribute-based access control for microservices-based applications using a service mesh. NIST. https://doi.org/10.6028/NIST.SP.800-204B
2. Devanbu, P., & Stubblebine, S. (2000). Software engineering for security: A roadmap. https://doi.org/10.1145/336512.336584
3. Ferraiolo, D., Kuhn, R., & Chandramouli, R. (2016). Role-based access control. Artech House. https://doi.org/10.1201/9781315369440
4. Ghotbi, S. H., & Fischer, B. (2013). Fine-grained role- and attribute-based access control for web applications. https://doi.org/10.1007/978-3-642-38709-8_12
5. Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006). Attribute-based encryption for fine-grained access control. https://doi.org/10.1145/1180405.1180418
6. Hu, V. C., Kuhn, D. R., Ferraiolo, D. F., & Voas, J. (2015). Attribute-based access control. Computer, 48(2), 85–88. https://doi.org/10.1109/mc.2015.33
7. Hu, V. C., Kuhn, D. R., & Ferraiolo, D. F. (2018). Access control for emerging distributed systems. Computer, 51(10), 100–103. https://doi.org/10.1109/mc.2018.3971365
8. Khare, S., Thakur, P., Talwandi, N. S., & Yadav, V. (2024). Securing microservice architecture: Load balancing and role-based access control. https://doi.org/10.63503/j.ijaimd.2024.7
9. Madkaikar, G., Sural, S., Vaidya, J., & Atluri, V. (2024). Queuing theoretic analysis of dynamic attribute-based access control systems. https://doi.org/10.1007/978-3-031-65175-5_23
10. Nehme, A., Jesus, V., Mahbub, K., & Abdallah, A. (2019). Fine-grained access control for microservices. https://doi.org/10.1007/978-3-030-18419-3_19
11. Newman, S. (2015). Building microservices. O'Reilly Media.
12. Salehi, S. A., Han, R., Rudolph, C., & Grobler, M. (2023). DACP: Enforcing a dynamic access control policy in cross-domain environments. Computer Networks, 237, 110049. https://doi.org/10.1016/j.comnet.2023.110049
13. Sänger, N., & Abeck, S. (2023). User authorization in microservice-based applications. Software, 2(3), 400–426. https://doi.org/10.3390/software2030019
14. Sandhu, R., Coyne, E., Feinstein, H., & Youman, C. (1996). Role-based access control models. https://doi.org/10.1109/2.485845
15. Sandhu, R., & Samarati, P. (1994). Access control: Principle and practice. https://doi.org/10.1109/35.312842
16. Schneider, M., Zieschinski, S., & Abeck, S. (2021). A test concept for microservice-based applications. https://doi.org/10.1109/sose52839.2021.00025
17. Sidler, J., Braun, E., Schmitt, C., Schlachter, T., & Hagenmeyer, V. (2022). Microservice-based architecture for integration systems. https://doi.org/10.1007/978-3-030-88063-7_3
18. Teerakanok, S., Uehara, T., & Inomata, A. (2021). Migrating to zero trust architecture: Reviews and challenges. https://doi.org/10.1155/2021/9947347
19. Wang, H., Chen, Y., & Xu, Z. (2023). Decentralized access control for secure microservices cooperation with blockchain. ISA Transactions, 141, 44–51. https://doi.org/10.1016/j.isatra.2023.07.018
20. Yuan, E., & Tong, J. (2005). Attribute-based access control (ABAC) for web services. https://doi.org/10.1109/icws.2005.25

FUNDING

The authors received no funding for the development of this research.

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest in this study and that the processes established by this journal have been ethically followed. Furthermore, they confirm that this work has not been published, either partially or in full, in any other journal.

AUTHORSHIP CONTRIBUTIONS

Conceptualization: DORG
Data curation: DORG
Formal analysis: DORG
Investigation: DORG
Methodology: DORG
Project administration: DORG
Resources: DORG
Software: DORG
Supervision: DORG
Validation: DORG
Visualization: DORG
Writing – original draft: DORG
Writing – review and editing: DORG