

Comparative Analysis of RBAC and ABAC Models in Modern Enterprise Systems

Вінницький національний технічний університет

Анотація

В статті проведено порівняльний аналіз двох домінуючих моделей контролю доступу: RBAC (на основі ролей) та ABAC (на основі атрибутів). Розглянуто архітектурні відмінності, переваги та обмеження кожного підходу в контексті побудови корпоративних систем захисту даних.

Ключові слова

Контроль доступу, RBAC, ABAC, інформаційна безпека, політики доступу, корпоративні системи, авторизація.

Abstract

This article presents a comparative analysis of two dominant access control models: RBAC (Role-Based Access Control) and ABAC (Attribute-Based Access Control). Architectural differences, advantages, and limitations of each approach are examined in the context of building enterprise data protection systems.

Keywords

Access Control, RBAC, ABAC, Information Security, Access Policies, Enterprise Systems, Authorization.

In the landscape of modern enterprise software development, ensuring the confidentiality and integrity of data is a paramount concern. As systems evolve from simple client-server applications to complex monolithic and distributed architectures, the mechanisms for granting user privileges and controlling access must also evolve. The core of this security infrastructure is the access control model. Currently, the industry relies heavily on two primary methodologies: Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC).

RBAC has established itself as the traditional standard for most corporate environments. The fundamental principle of this model is the segregation of duties based on organizational hierarchy. Instead of assigning permissions directly to users, administrators create "roles" (e.g., Manager, Auditor, Developer) that aggregate a specific set of privileges. Users are then mapped to these roles.

The primary strength of RBAC lies in its alignment with static organizational structures. It significantly simplifies the onboarding process: when a new employee joins a department, they are simply assigned the corresponding role. However, as noted by security researchers, RBAC suffers from a phenomenon known as "role explosion." In large enterprises, the necessity for minor exceptions leads to the creation of hundreds of specific roles (e.g., "Manager_DayShift_BuildingA"), making the system cumbersome and difficult to audit [1]. Furthermore, RBAC lacks context awareness; it typically cannot natively handle dynamic conditions such as "access allowed only during business hours".

To summarize the impact of this model, the key advantages of RBAC include:

- Streamlined Administration: Permissions are assigned to roles rather than individuals, significantly reducing manual overhead during personnel changes.
- Security Compliance: Users acquire only the permissions necessary for their specific job functions (Principle of Least Privilege).
- Scalability: The model adapts efficiently to standard organizational hierarchy changes.

However, the disadvantages are notable in complex environments:

- Role Explosion: As previously mentioned, the accumulation of highly specific roles can lead to management chaos.

- Lack of Context: The inability to account for dynamic variables (context) limits its effectiveness in high-security scenarios requiring conditional access.

To address the limitations of static roles, the Attribute-Based Access Control (ABAC) model was introduced. ABAC represents a shift from coarse-grained to fine-grained authorization. In this model, access decisions are dynamic and calculated at runtime based on policies that evaluate four key categories of attributes:

1. Subject attributes (who is requesting access: clearance level, department).
2. Resource attributes (what is being accessed: file sensitivity, owner).
3. Action attributes (what is being done: read, write, delete).
4. Environment attributes (context: time of day, IP address, device security status).

This approach allows for the implementation of complex Boolean logic policies, such as "Allow access if User.Department equals Resource.Department AND Time is between 9 AM and 6 PM" [2]. While ABAC offers unparalleled flexibility and granularity, it introduces serious implementation complexity. The processing overhead for evaluating complex policies for every request is higher than the simple lookup mechanism of RBAC. Additionally, ABAC requires a well-maintained database of attributes across the entire organization to function correctly in different environments [3].

Despite the complexity, ABAC offers superior advantages for modern systems:

- Extreme Granularity: Policies can be as specific as required, covering complex logic that roles cannot handle.
- Context Awareness: It natively incorporates environmental factors like time, location, and device status into the decision process.
- Adaptability: It suits dynamic and distributed environments where users and resources change frequently.

The main disadvantages that hinder widespread adoption include:

- Implementation Difficulty: Designing and maintaining a conflict-free policy set is intellectually demanding.
- Performance Overhead: Evaluating complex boolean logic for every request requires more computational resources compared to RBAC.
- Attribute Governance: The system is only as reliable as the data it uses; managing the integrity of user and resource attributes is a separate challenge.

Comparing the two models, it becomes evident that the choice depends on the specific requirements of the system. RBAC is efficient for systems with clearly defined, static user groups where coarse-grained access is sufficient. It is less resource-intensive and easier to visualize for audits. In contrast, ABAC is essential for environments requiring Zero Trust architecture, where context is crucial, and policies must adapt to rapidly changing situations without code adjustments.

In conclusion, while RBAC remains the workhorse of enterprise authorization due to its simplicity and manageability, modern security requirements are increasingly driving the adoption of ABAC elements. For many complex enterprise systems, a hybrid approach – using RBAC for high-level permissions and ABAC for specific, sensitive resource restrictions – often provides the optimal balance between security, performance, complexity and maintainability [4].

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. NIST (National Institute of Standards and Technology). (2016). Role-Based Access Control (RBAC) and Role Blowup. [Електронний ресурс]. URL: <https://csrc.nist.gov/projects/role-based-access-control>
2. Hu, V. C., Ferraiolo, D., Kuhn, R., et al. (2014). Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162. [Електронний ресурс]. URL: <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf>
3. Okta. (2023). RBAC vs. ABAC: What's the Difference? [Електронний ресурс]. URL: <https://www.okta.com/identity-101/rbac-vs-abac/>
4. OWASP Foundation. (2024). Access Control Methods and Models. [Електронний ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html

Поліщук Володимир Леонідович – студент групи ІКН-25м, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, Вінниця, e-mail: volodymyr8595@urk.net.

Мельник Олеся Дмитрівна – доцент кафедри іноземних мов, кандидат філологічних наук, Вінницький національний технічний університет, м. Вінниця, e-mail: prysyazhnalesya@gmail.com.

Polishchuk Volodymyr L. – student of Intelligent Information Technologies and Automation Department, Vinnytsia National Technical University, Vinnytsia, email: volodymyr8595@urk.net.

Melnyk Olesya D. – Associate Professor of the Department of Foreign Languages, Candidate of Philological Sciences, Vinnytsia National Technical University, Vinnytsia, e-mail: prysyazhnalesya@gmail.com.