

Access Control Models: DAC, RBAC

Majji Hari Krishna

Department of Computer Science & Engineering

Indian Institute of Technology Madras

Chennai, India

cs25m028@smail.iitm.ac.in

Abstract—Access control is a fundamental pillar of information security, ensuring that users can only access resources for which they are explicitly authorized. Among the myriad of access control models, Discretionary Access Control (DAC) and Role-Based Access Control (RBAC) are two of the most foundational and widely implemented. This report provides a comprehensive examination and comparative analysis of these two protocols. We begin by establishing the theoretical underpinnings of each model, detailing their core components, operational mechanisms, and formal representations. For DAC, we explore its owner-centric philosophy, the implementation through Access Control Lists (ACLs), and its inherent flexibility. For RBAC, we delve into its organization-centric approach, defining the relationships between users, roles, and permissions, and its alignment with the principle of least privilege. The report presents a detailed side-by-side comparison, highlighting critical differences in administration, scalability, security policy enforcement, and typical use cases. Through an analysis of real-world applications—from standard operating systems employing DAC to complex enterprise systems leveraging RBAC—this paper illustrates the practical strengths and weaknesses of each model. Furthermore, we discuss the challenges associated with each protocol, such as DAC’s susceptibility to Trojan horse attacks and RBAC’s potential for role explosion. The analysis concludes by summarizing the key differentiators and providing guidance on selecting the appropriate model based on an organization’s security requirements, size, and operational complexity.

Index Terms—Access Control, Information Security, DAC, RBAC, Access Control List (ACL), Least Privilege, Security Models.

I. INTRODUCTION

In the digital age, the protection of sensitive information is paramount. Organizations, governments, and individuals rely on computer systems to store, process, and transmit vast amounts of data, ranging from personal files to classified state secrets. Unauthorized access to this data can lead to catastrophic consequences, including financial loss, identity theft, and breaches of national security. The primary mechanism for preventing such unauthorized access is the implementation of a robust access control policy.

Access control is the selective restriction of access to a resource. It is the process by which a system grants or denies requests to access, use, or manipulate a resource based on a predefined security policy. The entities requesting access are typically users or processes, referred to as **subjects**. The resources being protected, such as files, databases, or network services, are referred to as **objects**.

Over the decades, several models have been developed to formalize and implement access control policies. These models

provide a framework for defining and enforcing security rules. Among the most prominent are Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC). While MAC is typically reserved for high-security environments like military and government systems, DAC and RBAC are widely used across the entire spectrum of computing, from personal devices to large-scale enterprise applications.

DAC is characterized by its flexibility, placing the control of access permissions in the hands of the resource owner. This model is intuitive and mirrors real-world ownership concepts. Conversely, RBAC takes a more structured, centralized approach, assigning permissions based on an individual’s role or job function within an organization. This model is designed to enforce the principle of least privilege and simplify administration in complex environments.

This paper aims to provide a detailed, comparative study of the DAC and RBAC protocols. We will dissect their architectures, formal models, advantages, and limitations. By examining their real-world applications and contrasting their core philosophies, this report will serve as a comprehensive guide for understanding two of the most critical technologies in modern cybersecurity. The structure of this paper is as follows: Section II lists relevant acronyms. Section III formalizes the models. Sections IV and V provide in-depth discussions on DAC and RBAC, respectively. Section VI offers a direct comparison. Section VII explores applications and use cases. Section VIII discusses limitations and future trends, followed by a conclusion in Section IX.

II. ABBREVIATIONS AND ACRONYMS

TABLE I
ABBREVIATIONS AND ACRONYMS

| Acronym | Description |
|---------|--|
| ACL | Access Control List |
| ABAC | Attribute-Based Access Control |
| CRM | Customer Relationship Management |
| DAC | Discretionary Access Control |
| ERP | Enterprise Resource Planning |
| IAM | Identity and Access Management |
| MAC | Mandatory Access Control |
| NIST | National Institute of Standards and Technology |
| OS | Operating System |
| PA | Permission-to-Role Assignment |
| RBAC | Role-Based Access Control |
| UA | User-to-Role Assignment |

III. FORMAL MODELS

To understand the precise behavior of access control systems, it is useful to define them with mathematical formalism.

A. Formalizing Discretionary Access Control (DAC)

The state of a DAC system can be described by an **access control matrix**, as proposed by Lampson and further developed by Harrison, Ruzzo, and Ullman (HRU model) [1]. This matrix, denoted as M , describes the rights of subjects over objects.

Let S be the set of subjects and O be the set of objects. Let R be the set of rights (e.g., read, write, execute, own). The access control matrix M is a two-dimensional array where the rows correspond to subjects and the columns correspond to objects. An entry $M[s, o]$ where $s \in S$ and $o \in O$, contains a subset of rights $r \subseteq R$.

$$M[s, o] = \{r_1, r_2, \dots, r_k\} \quad \text{where } r_i \in R \quad (1)$$

For example, if $M[\text{Alice}, \text{file.txt}] = \{\text{read}, \text{write}\}$, it means the subject Alice has read and write permissions for the object file.txt.

The "discretionary" aspect comes from rules that allow subjects to modify the matrix. For example, if 'own' $\in M[s, o]$, then subject s can execute commands to add or remove rights for another subject s' over object o .

$$\text{grant}(s, s', o, r) \implies \text{add } r \text{ to } M[s', o] \quad (2)$$

This formalization clearly shows how permissions are tied directly to subject-object pairs and how a subject with ownership rights can alter these pairings.

B. Formalizing Role-Based Access Control (RBAC)

The NIST standard for RBAC provides a widely accepted formalization [2]. The core RBAC model consists of several sets and relations:

- **U, R, P, S**: sets of users, roles, permissions, and sessions, respectively.
- **PA** $\subseteq P \times R$: a many-to-many permission-to-role assignment relation.
- **UA** $\subseteq U \times R$: a many-to-many user-to-role assignment relation.
- **user_sessions(u)**: a function mapping a user $u \in U$ to a set of sessions.
- **session_roles(s)**: a function mapping a session $s \in S$ to a set of roles.

A user can exercise a permission p if and only if that permission is authorized for one of the user's active roles.

$$\exists r \in \text{session_roles}(s) \text{ such that } (p, r) \in \text{PA} \quad (3)$$

This model introduces a layer of abstraction—the **role**—between users and permissions. A user is not directly granted permission; instead, they are assigned a role, and the role itself is what holds the permissions. This indirection is the key to RBAC's administrative scalability. Further extensions to the model include role hierarchies and constraints like Separation of Duty (SoD).

IV. DISCRETIONARY ACCESS CONTROL (DAC)

Discretionary Access Control is an access control policy determined by the owner of an object. The owner has the discretion to grant or revoke access permissions for that object to other subjects. This model is one of the simplest and most widely used, forming the basis for security in most commercial operating systems.

A. Core Principles and Implementation

The fundamental principle of DAC is **owner-controlled access**. When a subject creates a new object (e.g., a file or directory), they become its owner and are granted full control over it. This control is typically managed through an Access Control List (ACL).

An ACL is a data structure associated with an object that lists all subjects (users or groups) that have been granted access rights to it. Each entry in the ACL, known as an Access Control Entry (ACE), specifies a subject and the set of operations they are permitted to perform.

For instance, in a UNIX-like operating system, file permissions are a classic example of DAC. Each file has an owner and a group, and permissions are defined for three classes of subjects:

- 1) **Owner**: The user who created the file.
- 2) **Group**: A group of users who share permissions.
- 3) **Others**: All other users on the system.

Permissions are specified for reading ('r'), writing ('w'), and executing ('x'). A command like `chmod 754 file.txt` sets the owner's permissions to 'rwx' (read, write, execute), the group's to 'rx' (read, execute), and others' to 'r' (read). The owner can change these permissions at any time.

B. Advantages of DAC

- 1) **Flexibility and Ease of Use**: DAC is highly flexible. Owners can share resources with collaborators quickly and dynamically without needing to go through a central administrator. This fosters a collaborative environment.
- 2) **Simplicity**: The concept of ownership is intuitive and easy for end-users to understand and manage for their own files.
- 3) **Granularity**: Permissions can be assigned on a per-user, per-object basis, allowing for very fine-grained control if needed.

C. Disadvantages of DAC

- 1) **Susceptibility to Trojan Horses**: DAC's major vulnerability lies in its handling of program execution. If a user runs a program (a Trojan horse) that has malicious code, that program inherits all of the user's permissions. It can then perform unauthorized actions on behalf of the user, such as deleting files or copying sensitive data to an unauthorized location, without the user's knowledge [3].
- 2) **Lack of Centralized Control**: Because permissions are managed by individual users, it is difficult to enforce a consistent, organization-wide security policy. This can

lead to security gaps and inconsistent permission settings across the system.

- 3) **Administrative Overhead:** In a large system, tracking and managing permissions can become incredibly complex. Revoking access for a user who has left the organization can be a tedious and error-prone process, as an administrator must check the ACL of every object they might have had access to.

V. ROLE-BASED ACCESS CONTROL (RBAC)

Role-Based Access Control is a policy-neutral access control model that has gained widespread acceptance as a best practice for managing permissions in medium-to-large-scale enterprises. In RBAC, access decisions are based on the roles that individual users have as part of an organization.

A. Core Principles and Implementation

The central idea of RBAC is to introduce a level of indirection between users and permissions. Rather than assigning permissions directly to users, permissions are assigned to roles, and users are then assigned to roles. A role is a semantic construct that represents a job function or title within an organization (e.g., 'Accountant', 'HR Manager', 'Database Administrator').

The implementation of RBAC involves three primary mappings:

- 1) **Permission-Role Assignment (PA):** Specific system permissions (e.g., 'create-invoice', 'view-salary-data', 'reboot-server') are assigned to the relevant roles.
- 2) **User-Role Assignment (UA):** Users are assigned to one or more roles based on their job responsibilities.
- 3) **Session Management:** When a user logs in, they create a session and can choose to activate a subset of the roles they are assigned. This allows them to operate with only the permissions necessary for the immediate task, adhering to the principle of least privilege.

This structure simplifies security administration. When a new employee joins, the administrator simply assigns them the appropriate role(s). When an employee's job changes, their role assignment is updated, and their permissions change automatically. When an employee leaves, their role assignments are simply revoked, effectively and cleanly severing all access.

B. Advantages of RBAC

- 1) **Simplified Administration:** Managing roles is far simpler than managing individual permissions for thousands of users. This significantly reduces administrative workload and the potential for error [4].
- 2) **Enforcement of Least Privilege:** RBAC makes it easy to implement the principle of least privilege. By defining roles with the minimal set of permissions required for a job function, organizations can reduce the risk of users having excessive, unnecessary access rights.
- 3) **Scalability:** RBAC scales exceptionally well. As an organization grows, new users are simply slotted into

existing roles. New job functions can be accommodated by creating new roles.

- 4) **Policy Compliance:** RBAC helps organizations meet regulatory and statutory requirements (e.g., HIPAA, SOX) that mandate privacy and confidentiality, as it allows for systematic enforcement and auditing of access policies.

C. Disadvantages of RBAC

- 1) **Initial Complexity:** The initial implementation of RBAC can be a complex and resource-intensive project. It requires a thorough analysis of the organization's business processes and job functions to define a logical and effective set of roles.
- 2) **Role Explosion:** In large, complex organizations, the number of distinct roles can become unmanageably large. This "role explosion" can reintroduce the complexity that RBAC was designed to solve. Careful role engineering and management are required to prevent this.
- 3) **Static Nature:** Core RBAC models can be static. They do not easily account for contextual attributes, such as time of day, user location, or the state of a resource, which may be required for more dynamic access control decisions.

VI. COMPARATIVE ANALYSIS: DAC VS. RBAC

While both models aim to secure resources, their philosophical and practical differences are stark. The choice between them depends heavily on the context of the system being secured.

TABLE II
COMPARISON OF DAC AND RBAC CHARACTERISTICS

| Feature | Discretionary Access Control (DAC) | Role-Based Access Control (RBAC) |
|----------------------------|---|---|
| Control Loc | Decentralized; at the discretion of the resource owner. | Centralized; managed by system administrators based on organizational policy. |
| Access Basis | Based on the identity of the subject (user or group). | Based on the assigned role(s) of the subject. |
| Primary Mechanism | Access Control Lists (ACLs). | Role-Permission and User-Role assignments. |
| Flexibility | Very high; permissions can be changed dynamically by users. | Moderate; requires administrative action to change roles or role permissions. |
| Scalability | Poor; becomes unmanageable in large organizations. | Excellent; scales efficiently with organizational growth. |
| Security Principle | Focuses on ease of sharing and collaboration. | Enforces the principle of least privilege and separation of duty. |
| Policy Enforcement | Inconsistent; depends on the diligence of individual users. | Consistent; enforces a single, organization-wide security policy. |
| Typical Environment | Personal computing, small workgroups, academic projects. | Enterprise systems (ERP, CRM), government, health-care, financial institutions. |

The most fundamental difference lies in how permissions are managed. In DAC, permissions are attached to an object, and control flows from the object's owner. In RBAC, permissions are attached to a role, and control flows from a central administrative policy that maps users to these roles.

This leads to a trade-off: DAC offers unparalleled flexibility for the end-user, making it ideal for environments where information sharing is fluid and dynamic. However, this comes at the cost of centralized oversight and policy enforcement. RBAC enforces a rigid, centrally managed policy, which is less flexible for the end-user but provides superior security, auditability, and administrative efficiency for an organization.

VII. USES AND REAL-LIFE APPLICATIONS

The theoretical differences between DAC and RBAC are best understood through their practical implementations in real-world systems.

A. Applications of DAC

- 1) **Desktop Operating Systems:** The most common examples of DAC are found in file systems of OSs like Microsoft Windows (NTFS permissions), macOS, and Linux. When you create a document on your computer, you are its owner. You can then right-click the file, go to 'Properties' or 'Get Info', and set specific permissions for other users or groups on that computer or network.
- 2) **Social Media Platforms:** On platforms like Facebook or Google Drive, when you upload a photo or create a document, you can often choose who gets to see it—"Only Me," "Friends," or "Public." This is a form of DAC, where you, the owner of the data object, are making discretionary decisions about its access.
- 3) **Small-Scale Collaborative Tools:** Tools like Trello or shared folders in Dropbox often use a DAC-like model. The creator of a board or folder can invite specific collaborators and assign them different levels of access (e.g., view-only, editor).

B. Applications of RBAC

- 1) **Enterprise Resource Planning (ERP) Systems:** In a large company using a system like SAP or Oracle, thousands of employees need access. An employee in the finance department is assigned the 'Accountant' role. This role automatically grants them access to view financial ledgers, create invoices, and process payments, but not to view HR records or modify production schedules.
- 2) **Cloud Computing Platforms:** Services like Amazon Web Services (AWS) and Microsoft Azure rely heavily on RBAC through their Identity and Access Management (IAM) systems. An administrator can create roles like 'Developer', 'DatabaseAdmin', or 'Auditor'. A user assigned the 'Developer' role can spin up and manage virtual machines but cannot change billing information or access sensitive security logs, which are permissions reserved for other roles.

- 3) **Healthcare Information Systems:** To comply with regulations like HIPAA, hospitals use RBAC to control access to Electronic Health Records (EHR). A doctor is assigned a 'Physician' role, allowing them to view and modify the records of patients under their care. A 'Billing Clerk' role allows access only to the insurance and payment information associated with a patient, not their clinical diagnoses or treatment history. This ensures that employees only access the minimum necessary information required to perform their jobs.

VIII. CHALLENGES, HYBRID MODELS, AND FUTURE TRENDS

Neither DAC nor RBAC is a perfect solution, and both face challenges. Modern systems often employ hybrid models or look towards more advanced paradigms.

A. Limitations and Challenges

As discussed, DAC's primary weakness is its lack of central control, which can lead to security vulnerabilities. RBAC's main challenge is managing complexity, particularly avoiding **role explosion**. This occurs when an organization creates too many granular roles, making the system as difficult to manage as a large-scale DAC implementation. Another RBAC challenge is managing **toxic combinations**, where a user assigned multiple roles inadvertently gains a set of permissions that could lead to fraud (e.g., being able to both create and approve a payment). This requires careful implementation of Separation of Duty (SoD) constraints.

B. Hybrid Models

Many systems implement a hybrid approach. A Linux system, for example, uses DAC for its standard file permissions. However, enhancements like SELinux (Security-Enhanced Linux) can be layered on top to enforce a system-wide Mandatory Access Control (MAC) or Role-Based Access Control (RBAC) policy, overriding the discretionary permissions set by users. This allows for both user flexibility at a low level and strong, centralized security policy at a high level.

C. Future Trends: Attribute-Based Access Control (ABAC)

The successor to RBAC is widely considered to be **Attribute-Based Access Control (ABAC)** [5]. ABAC, also known as Policy-Based Access Control (PBAC), makes access decisions based on a combination of attributes of the subject, the object, the requested action, and the environment.

For example, an ABAC policy might state: "Allow doctors (subject attribute) to access medical records (object attribute) of their own patients (relational attribute) only during work hours (environmental attribute) from a hospital-owned device (environmental attribute)." This provides a far more dynamic and context-aware level of control than the relatively static roles in RBAC, and it is becoming the standard for modern, fine-grained access control, especially in Zero Trust architectures.

IX. CONCLUSION

Discretionary Access Control and Role-Based Access Control represent two distinct and highly influential philosophies in the field of information security. DAC, with its owner-centric, flexible model, empowers individual users and fosters collaboration, making it a natural fit for personal computing and small-scale environments. Its strengths lie in its simplicity and intuitiveness. However, this same flexibility becomes a liability in larger, structured environments, where it fails to provide the centralized control and policy consistency necessary for robust security.

RBAC addresses these shortcomings by introducing a layer of abstraction through roles. By aligning permissions with organizational job functions, RBAC delivers a scalable, manageable, and secure framework that effectively enforces the principle of least privilege. It has rightfully become the de facto standard for access control in the enterprise world, enabling organizations to meet complex security and compliance demands.

The choice between DAC and RBAC is not a matter of one being universally superior to the other, but rather a decision contingent on the specific requirements of the system. For unstructured data and dynamic collaboration, DAC excels. For structured processes and enterprise security, RBAC is indispensable. As the digital landscape evolves, these foundational models will continue to be relevant, even as they are augmented and sometimes superseded by more context-aware paradigms like ABAC, which build upon the lessons learned from their predecessors. A thorough understanding of both DAC and RBAC remains essential for any cybersecurity professional tasked with designing and implementing effective access control systems.

ACKNOWLEDGMENT

The author would like to thank [Professor's Name/Course Name], of the Department of Computer Science & Engineering at the Indian Institute of Technology Madras, for their guidance and invaluable feedback throughout the development of this course project.

REFERENCES

- [1] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461-471, Aug. 1976.
- [2] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed. Norwood, MA: Artech House, 2007.
- [3] W. E. Boebert and R. Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies," in *Proceedings of the 8th National Computer Security Conference*, 1985, pp. 18-27.
- [4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38-47, Feb. 1996.
- [5] V. C. Hu, D. F. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST Special Publication 800-162, Jan. 2014.