Written Report on Access Controls

Rafael Enrico C. Hugo
De La Salle University
Manila, Philippines
rafael_hugo@dlsu.edu.ph

I. Introduction

Security is a fundamental aspect of web applications and web development. As technology advances and the digital world becomes more integral to business operations, ensuring the protection of data, systems, and resources against unauthorized access is becoming more important. Breach of security can lead to significant financial losses and legal repercussions, making the implementation of security measures a necessity in this day and age.

One of the core components of security in web applications is access control. An access control, as according to Microsoft, is a security mechanism responsible for determining an individual's permissions on a specific set of data or resources. It helps regulates who is allowed to access information, when can they do so, and how they can manage or manipulate it. The implementation of access controls is reliant on authentication and authorization. Authentication precedes authorization as it first needs to verify the user's identity. Once the user is verified, the authorization will determine the level of access granted to them based on predefined permissions and roles.

There is an emphasis on predefined permissions and roles. In the context of web development, a user should be given access based on only what they need. This concept is known to be the Principle of Least Privilege. Different users are given different base permissions, usually based on their roles which should already be established prior to the development of the web application.

The access control acts as a security mechanism. As such, it helps keep confidential information from being stolen by unauthorized users. These users can range from external threat actors, like hackers and cybercriminals, to internal personnel, including organization's employees. Limiting an individual's access prevents essential data breaches and unwanted resource utilization, as in the cases of compute power, bandwidth, or storage space. Without an effective access control, web applications expose themselves to vulnerabilities that could lead to operational consequences.

Aside from protecting sensitive data, an access control is essential to complying with an organization's or country's regulatory and security policies. Industries such as healthcare, finance, and government adhere to quite strict regulations designed to safeguard user data and to ensure ethical data management practices. For example, healthcare organizations in the United States must comply with the Health Insurance Portability and Accountability Act (HIPAA). In Europe, the General Data Protection Regulation (GDPR) enforces privacy laws governing the collection and processing of personal data. In the context of the Philippines, individuals and organizations alike must adhere to the Data Privacy Act of 2012 (DPA 2012) which is regulated by the National Privacy Commission. Moreover, the DPA 2012 establishes the legal standards for handling and securing personal information. In the case of non-compliance with these regulations, individuals and organizations are subject to substantial fines and legal liabilities.

An access control can be implemented in many different ways and so, there are different types of access controls. This paper will primary focus on three models: Capability-based access control, Role-based access control, and Discretionary access control.

II. Individual Model Analysis

The choice of access control models greatly depends on the organization's specific security requirements, operational needs, and regulatory obligations. Each model has its own unique advantages and has its own different use cases.

In order to delve deeper on each of the aforementioned access control models, the main talking points must first be established. This section is the paper is primarily concerned with an access model's formal definitions, core components, strengths, weaknesses, and use cases. Both a real-world and technical example will also be given to further define and also perhaps to serve as a visual for the reader.

A. Capability-based access control

A Capability-based access control (CapBAC) is an access control model that regulates permissions based on unique tokens or capabilities. These capabilities can be tokens or objects that explicitly define what actions a user can perform

on a resource. A token can be defined as a computer-generated code that can act as a verification tool of a user. Other than providing authentication, it can also authorize in the context of CapBAC. A capability on the other hand can be defined as a pair, an object and a set of permissions. Granting permissions in CapBAC is usually not based on the user's identity but on the possession of the token or capability.

In the context of Internet of Things (IoT), a visitor can be given a temporary password to a smart door lock for a specific time period. A visitor's access to the house is dependent on whether they know or have the details of the temporary password. This visitor need not be recognized as anyone specific.

CapBAC can also be seen in web applications that make use of API keys and API authentication mechanisms. Other internet protocols like OAuth tokens can also grant access to external applications or users. A common example of a cloud provider is Amazon Web Services and their feature AWS S3 allows users to generate access tokens that can grant other users permissions. Anyone can access certain features based on whether they have the token or capability for it.

This method of access control greatly reduces the reliance on identity-based authentication. It also enforces the Principle of Least Privilege given that the token will dictate what actions can be done. Greater control is allowed since access can have well-defined restrictions. The time to access data and resources can also be set to limited, therefore giving the token an expiration date.

Since the authentication process has less importance in CapBAC, anyone having access to a token or capability can then share this to external threat actors. While these tokens can expire, the risk of granting access to unwanted individuals can still vary on the management of the tokens.

Cloud computing primarily makes use of CapBAC as it would need API keys and tokens for users to access such services. The strain on authentication processes is non-existent and it ensures that the authorization processes limit the actions of a user based on the token. IoT can also make use of this model as proposed by Domenico Rotondi and Salvatore Piccione.

B. Role-based access control

A Role-based access control (RBAC) is an access control model that regulates access based on a user's predefined role. This role is usually already part of an organization and then integrated into the application. Each role already has a predefined set of permissions.

A real-world example for a role-based access control is the following: Imagine an educational institution with three roles: student, teaching personnel, and non-teaching personnel. Students may only access course materials, modules, and submit assignments while teaching personnel have the ability to modify course content and to evaluate a student's performance. Non-teaching staff cannot do either but they may be in charge of maintaining the environment of the institution.

A more technical example of a role-based access control can be seen in corporate software applications with different levels of user access. This web application may have roles such as employee, manager, and administrator. Employees can only view their personal information like salary while managers have additional permissions to evaluate team performance. Administrators, on the other hand, have control over the system and user management. There is a sort of hierarchical structure in this access control which should ensure that users only have access to the information necessary for their role.

One of the primary strengths of RBAC is its ability to implement the Principle of Least Privilege. Restricting users to only permissions related to their role, security risks can be minimized. RBAC simplifies management by allowing only administrators to assign permissions at the role level rather than a per-user basis. This allows for flexibility as it would be more time consuming to grant access one by one each user. Moreover, this approach helps with compliance with regulatory requirements as access control can be clearly defined and audited.

While RBAC is great for simplifying user management, it has its own limitations. One of the main challenges is initial setup and maintenance. As earlier discussed, this access control must already have predefined roles and role permissions. These cannot be an afterthought. Other than strict planning, organizations with dynamic access such as two people of the same role having different permissions greatly limits the capabilities of RBAC. Larger organizations can also lead to an incident called role explosion. It is when the number of roles increase to accommodate different permissions between individuals.

Despite these weaknesses, RBAC remains a widely used access control due its structured approach and effectiveness in managing permissions. Many industries like healthcare, finance, and education rely on RBAC given how there is a clear hierarchy within these industries.

C. Discretionary access control

A Discretionary access control (DAC) is an access control model that grants or restricts access on the choice or discretion of the owner or manager. DAC does not rely on predefined roles rather DAC allows users to define who can access their data and resources and what access level they have. It is a case-by-case control over resources.

To illustrate a real-world example of DAC, we can look towards an office setting where employees store documents on a shared network drive. An employee that creates a document can choose to either share it to specific coworkers or to keep it private. Since the document is owned by that employee, they have the discretion for access over the file. This access can range from viewing, commenting, or editing the file itself.

On a more technical level, DAC is commonly and more evidently seen in operating systems such as Windows and Linux. Each file or directory has an associated owner who can determine the access levels for other users. In Linux, there is a command "chmod" which allows assignment of permissions such as read, write, and execute. Windows offers a more user interface friendly approach through the ACLs guided user interface (GUI).

DAC is quite flexible and it is one of its strengths. It allows owners to make the decisions about access to facilitate collaboration and ease of use. Given that there are no predefined roles, its implementation thrives on user-centric environments where there is a clear owner for a data or resource. It is also more commonly seen in collaborative workspaces where individuals frequently share and manage their own data.

Although it is flexible, it is weak in the security management side. Since the owners have the ability to assign permissions, the risk of data exposure relies heavily on them. An external threat actor may unknowingly be granted access which could lead to data breaches. DAC does not enforce strict organizational policies and this can pose compliance risks in industries with strict data protection regulations.

DAC's ease of implementation allows it to be a widely used access control. As given in an earlier example, operating systems and even platforms like Google drive have DAC implemented within them. Outside of those examples, it is also seen in smartphones where applications need permissions for the owner's data.

III. Comparison and Contrast

As explained in an earlier part of the paper, each access control model has unique characteristics that define how permissions are assigned and managed. RBAC, DAC, and CapBAC offer different approaches. This section of the paper deals with the similarities and differences of the models based on these criteria: access basis, permission assignment, security risks, environment, and access auditing.

Briefly explaining each criterion, access basis refers to where the access control was based on, permission assignment with how permission is delegated among users. Security risks tells of the possible vulnerabilities within each model. Environment deals with where this model is suited for. And lastly, access auditing which refers to revoking and logging of access.

A. Access basis

1) RBAC: is structured around predefined roles and predefined permissions. This role is tied to the organization and should be existent prior to the creation of the web application.

2) DAC: is identity-based, where an owner has authority to grant permissions. Unlike RBAC, DAC does not need roles rather only permissions and these permissions are granted through the owner or administrator.

3) CapBAC: on the other hand focuses on token-based access, where users are granted permissions through tokens. There is no need for roles or authentication in CapBAC.

B. Permission assignment

1) RBAC: centers around the organization's roles. It is quite rigid in granting permission as permissions are only granted by role assignment. It is not possible to individually change permissions without manipulating the permission of the whole role.

2) DAC: relies on an owner or administrative user to assign permissions manually. Given that they the owners have the responsibility to assign, they are also responsible for rescinding or revoking a user's access.

3) CapBAC: have permissions stored within the tokens through cryptography. The tokens containing the allowed permissions can be unique per user. Delegation in CapBAC is much more liberal than RBAC but still more rigid than DAC.

C. Security risks

1) RBAC: is rigid in its implementation therefore it security risks with RBAC is low. The risk of unauthorized access is also low as a user's access depends on their role. Despite the low security risk, it is not flexible when it comes to changes in security policies.

2) DAC: depends heavily on the owner or administrator of a resource. Its security is tied to how liberal the owner gives access and who they give access to. Without any form of rigidity, DAC is prone to unauthorized access and elevated privilege.

3) CapBAC: has some rigidity while offering security. The rigidity and all CapBAC security risks depend on how tokens are managed. Unencrypted tokens may be leaked and therefore may lead to unauthorized access, but this access is only limited to the permissions allowed within the token.

D. Environment

1) RBAC: is suited for structured organizations with predefined roles. This makes it simpler to manage access within the users.

2) DAC: is suited for collaborative applications and operating systems that support multiple users like Google documents or Windows respectively.

3) CapBAC: is suited for distributed systems like cloud-based services, microservice architectures like in the financial sector.

E. Access auditing

1) RBAC: ease of revoking access is simply implemented by removing a user from a specific role. It is still a manual process but still follows a structured approach.

2) DAC: ease of revoking access is on a case-by-case basis. It is manual and all permissions must be set and revoked individually.

3) CapBAC: has the ability to limit the access time. Access is more dynamic in CapBAC due to how tokens are configured.

Below is a summary of the similarities and differences between the three discussed access control models:

TABLE I
SUMMARY TABLE OF ROLE-BASED ACCESS CONTROL, DISCRETIONARY ACCESS CONTROL, AND CAPABILITY-BASED ACCESS CONTROL

Aspect	RBAC	DAC	CapBAC
Access Basis	Role-based, predefined roles	Identity-based, owner grants permissions	Token-based, access granted through tokens
Permission Assignment	Granted via role assignment	Owner manually assigns permissions	Tokens store permissions
Security Risks	Low	High	Medium
Environment Suitability	Structured organizations with predefined roles	Collaborative applications, multi-user OS environments	Distributed systems, cloud services
Access auditing	Removing user from role revokes all permissions	Manual revocation on a case-by-case basis	Dynamic revocation using time-limited tokens

IV. CONCLUSION

Each access control model is meant for different environments, systems, and applications. Their effectiveness is only as great as its implementation and if it catered to the application. For structure and security, RBAC is the way to go as its rigidness perfectly captures the hierarchical and role-based permissions. Although it does lack in the aspect of change or upgrade. DAC on the other hand is too reliant on the administrator but nevertheless, the full control an administrator has is also its strong point as it does not constrict access to roles. CapBAC looks to be the most optimal in the case for today's web applications. Tokens guarantee safe transactions and access especially when they are given an expiration date. Its security is also better than DAC as tokens encrypyted.

Choosing the right model depends on what the application needs based on security and flexibility.

REFERENCES

[1] Microsoft, "What is Access Control?", Microsoft Security, 2023. [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-access-control.

[2] NordLayer, "What is Access Control? Definition and Key Concepts", 2023. [Online]. Available: https://nordlayer.com/learn/access-control/what-is-access-control/.

[3] Cornell University, "Lecture 8: Access Control Models", 2005. [Online]. Available: https://www.cs.cornell.edu/courses/cs513/2005fa/L08.html.

[4] IBM, "What is Role-Based Access Control (RBAC)?", 2024. [Online]. Available: https://www.ibm.com/think/topics/rbac.

[5] Delinea, "Understanding Access Control Models and Methods", 2023. [Online]. Available: https://delinea.com/blog/access-control-models-methods.

[6] Permify, "What is Role Explosion? Understanding RBAC Limitations", 2023. [Online]. Available: https://permify.co/post/role-explosion/.

[7] Startup House, "What is Capability-Based Security?", 2023. [Online]. Available: https://startup-house.com/glossary/what-is-capability-based-security.

[8] Y. He, "RBAC and CapBAC in IoT", Dev.to, 2023. [Online]. Available: https://dev.to/yongchanghe/rbac-and-capbac-in-iot-37f3.

[9] Cloudflare, "Token-Based Authentication Explained", 2023. [Online]. Available: https://www.cloudflare.com/learning/access-management/token-based-authentication/.

[10] GeeksforGeeks, "How Does Token-Based Authentication Work?", 2023. [Online]. Available: https://www.geeksforgeeks.org/how-does-the-token-based-authentication-work/.

[11] D.Rotondi S.Piccione, "Managing Access Control for Things: a Capability Based Approach," in *Proceedings of the International Conference on Body Area Networks (BodyNets)*, 2012. DOI: 10.4108/icst.bodynets.2012.250234.