Access Control

Data Security and Privacy @ uninsubria

Roberto Vicario

2024/2025

Contents

1 Introduction

…

2 Mandatory Access Control (MAC)

…

2.1 Bell and La Padula Model

…

3 Discretionary Access Control (DAC)

…

3.1 Grant-Revoke Model

…

4 Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is a security model that assigns permissions to users based on their roles within an organization. In RBAC, users are assigned to roles, and each role has specific permissions associated with it. This model simplifies access management by grouping users into roles, making it easier to manage permissions at scale.

4.1 Key Components

Component	Description
Users	Individuals or entities that access the system.
Roles	Named groups of permissions that can be assigned to users. Roles represent job functions or responsibilities within the organization.
Permissions	Specific access rights to perform actions on resources. Permissions are associated with roles rather than individual users.
Sessions	Temporary associations between users and roles. A user can activate a role during a session to perform specific tasks.

4.2 Role Hierarchy

> **Definition:** *Role hierarchy* is a feature of RBAC that allows roles to inherit permissions from other roles. This simplifies permission management by allowing administrators to define a hierarchy of roles, where higher-level roles inherit permissions from lower-level roles.

More formally, a *role hierarchy* can be mathematically formalized as a partial order $RH \subseteq ROLES \times ROLES$ on the set of roles $ROLES$. The partial order is denoted by the dominance relation $\geq$:

- When $r_1 \geq r_2$, we say that r_1 is *senior* to r_2, and r_2 is *junior* to r_1.

4.2.1 Semantic Role Inheritance

Semantic	Description
User Inheritance	All users authorized for a role r are also authorized for any role r' where $r \geq r'$.
Permission Inheritance	A role r is authorized for all permissions for which any role r', such that $r \geq r'$, is authorized.

4.3 Separation of Duties (SoD)

> **Definition:** *Separation of Duties (SoD)* is a security principle that divides responsibilities among multiple users to prevent fraud and errors. It ensures that no single user has complete control over a critical process, reducing the risk of unauthorized actions.

4.3.1 Types of SoD

Type	Description
Static	Conflicts are resolved during system design.
Dynamic	Conflicts are resolved at runtime based on user actions.

5 SQL Access Control

SQL access control is a mechanism used to manage and restrict access to database resources based on user roles and permissions. It ensures that only authorized users can perform specific actions on the database, such as querying, updating, or deleting data.

5.1 SQL Injection

SQL injection is a code injection technique that exploits vulnerabilities in an application's software by injecting malicious SQL code into a query. This can lead to unauthorized access to sensitive data, data manipulation, or even complete control over the database.

> **Example:** Consider a web application that allows users to log in using their username and password. The application constructs a SQL query to authenticate users:

```sql
SELECT * FROM users WHERE username = ? AND password = ?;
```

If the application does not properly validate or sanitize user input, an attacker could inject malicious SQL code into the username or password fields. For example, if the attacker enters the following input for the username:

```sql
' OR 1 = 1; --
```

The resulting SQL query would be:

```sql
SELECT * FROM users WHERE username = '' OR 1 = 1; --' AND password = ?;
```

This query would always return true, allowing the attacker to bypass authentication and gain unauthorized access to the application.

5.1.1 SQL Injection Types

Type	Description
Classic	The attacker injects malicious SQL code into a query, allowing them to manipulate the database. This can include bypassing authentication, retrieving sensitive data, or modifying data.
Union-based	Combines results from multiple queries using the UNION operator to extract data from other tables.
Time-based	Exploits response delays to infer database information by observing query execution time.
Blind	Exploits application behavior to infer database information without direct feedback.

5.1.2 Mitigation Techniques

Technique	Description
Parameterized Queries	Use prepared statements to ensure user input is treated as data, not code.
Input Validation	Ensure user input is sanitized and conforms to expected formats to prevent malicious code execution.
Least Privilege	Grant only the necessary privileges to users and applications. Avoid using admin accounts for routine tasks.
Web Application Firewalls (WAFs)	Use WAFs to detect and block SQL injection attempts.
Object-Relational Mapping (ORM) Frameworks	Use ORM frameworks to simplify database interactions and mitigate SQL injection risks.