Access Control Models Implementation

- Each file contains 3 sections of codes. The first part is for the policy generator, the second part is for printing CSV files and the third section is to clear all the CSV files.
- Policy generator is designed in such a way that it creates one policy at a time but it can be easily modified to generate more policies.
- All the scenarios are manually crafted based on identifying the co-relations in the TPCH database and roles are based on realistic industry roles

This repository contains implementations of three different access control models: Attribute-Based Access Control (ABAC), Policy-Based Access Control (PBAC), and Role-Based Access Control (RBAC). These implementations are designed to simulate various security scenarios to demonstrate how different access control strategies can be applied.

File Descriptions

ABAC.ipynb

Description: Implements the Attribute-Based Access Control (ABAC) model. This notebook includes a Python class that simulates access control decisions based on user attributes, environmental conditions, and resource requirements.

Key Features:

Dynamic access control based on multiple attributes.

Simulation of environmental factors influencing access decisions.

Data logging to CSV files for access decisions and policies.

PBAC (1).ipynb

Description: Demonstrates Policy-Based Access Control (PBAC), focusing on the enforcement of complex organizational policies using Python.

Key Features:

Definition and enforcement of granular access policies.

Evaluation of access requests against predefined policies.

Detailed policy management and decision logging.

RBAC (2).ipynb

Description: Implements Role-Based Access Control (RBAC) where access permissions are assigned to roles rather than individual users, simplifying management and scalability.

Key Features:

Role definition and role assignment mechanisms.

Simulated access control decisions based on user roles.

Role hierarchies and constraints for comprehensive access control.

Usage Instructions

Prerequisites:

Python 3.x

Libraries: faker, csv, and other dependencies as needed.

Jupyter Notebook or an equivalent Python environment to run .ipynb files.

Running the Notebooks:

Open each notebook in Jupyter Notebook or your preferred environment.

Execute the cells in order to see the simulation of the access control model described within each notebook.

Regarding the scenarios

1. Comprehensive Attribute Design (ABAC)

Dynamic Attributes: The use of dynamically set environmental attributes like current_time, current_date, and network_security_level adds a layer of realism that reflects real-world conditions. This allows the system to mimic real-life scenarios where access decisions may depend on time-sensitive or context-sensitive information.

Diverse User and Resource Attributes: Attributes such as user_clearance, business_unit, data_sensitivity, and specific business-related attributes (e.g., market_segment, customer_revenue) allow the simulation to address a wide range of business roles and data access requirements. This diversity ensures that the model can handle complex decision-making processes where different users have varying levels of access based on their roles and the sensitivity of the data.

2. Role Hierarchies and Permissions (RBAC)

Role-Based Access Control: By defining specific roles such as financial_analyst, compliance_officer, inventory_manager, etc., and associating these roles with specific access permissions, the RBAC model realistically simulates organizational structures and access policies. This not only simplifies the management of permissions but also closely aligns with how many real organizations operate.

Role Hierarchies: The inclusion of scenarios that may imply role hierarchies (e.g., roles with overlapping permissions or subordinate roles) demonstrates a nuanced approach to access control, reflecting the layered access controls found in larger organizations.

3. Policy Constraints and Flexibility (PBAC)

Policy Evaluation: In environments where policies dictate access (suggested by the scenarios in PBAC), your implementation can flexibly evaluate access based on complex conditions beyond just roles or attributes. This might include evaluating combinations of attributes, environmental conditions, and even past access decisions or user behaviors.

Granular Control: Policies might enforce restrictions or grant access based on a combination of factors, such as time of day combined with the type of transaction and user location. This granularity is vital in sectors like finance or health where data sensitivity is paramount.

4. Environmental and Contextual Sensitivity

Real-time Conditions: By incorporating real-time conditions like current_time or operational_hours, the system can dynamically adjust to context, such as allowing access only during business hours or providing access based on current workload or system status (system_load).

5. Scenario Simulation and Testing

Wide Coverage: The scenarios cover a broad spectrum of typical business operations—financial reviews, compliance audits, market analysis, and more—offering a comprehensive test bed for the access control system. This broad coverage ensures that the access control models are robust and versatile.

Interactive Testing: By allowing interactive input to continue or stop scenario simulations, the system offers a hands-on feel for how changes in attributes or policies might affect real-time access control decisions.