Project Documentation: Access Control and Policy Enforcement System

Overview

This project implements a role-based access control (RBAC) system for managing user access to various resources in a healthcare environment. The system incorporates a **break-glass mechanism**, **zero-trust verification**, and **policy enforcement** to ensure security and compliance. The system is designed to regulate user access to critical healthcare data such as EMR (Electronic Medical Records), pharmacy data, lab results, and more.

Key Components

1. **Roles and Permissions**: Different user roles (e.g., doctors, nurses, admins) are associated with specific permissions (e.g., read, update, delete) to regulate their access to resources.
2. **Policy Templates**: Policies define the conditions under which users are permitted or denied access to resources. Policies are based on roles and attributes such as user shift status and device/location checks.
3. **Break-Glass Mechanism**: Allows emergency access to resources by overriding standard policy restrictions. This mechanism is logged for future auditing.
4. **Zero-Trust Verification**: Ensures that access is granted only under strict security conditions, such as active shift status, device authenticity, and location verification.

Modules and Files

1. **main.py**: Core script for simulating user access to resources with policy evaluation and enforcement. It includes:
    - Role-based policy evaluation using the PDP (Policy Decision Point).
    - Enforcement of decisions using the PEP (Policy Enforcement Point).
    - Break-glass mechanism for emergency access.
2. **roles.py**: Defines various user roles and the actions they are authorized to perform on resources.
    - Roles like "doctor", "nurse", "admin", etc., with specific permissions for each role (e.g., "read", "update").
3. **policies.py**: Contains access control policies for each role. Policies check whether a user has the appropriate role and attributes to perform a specific action on a resource.
4. **users.py**: Defines user attributes such as their role, active shift status, device ID, and location.
5. **access_logs.txt**: Logs every access attempt, including the user, resource, action, decision, and reason, for auditing and compliance purposes.

Detailed Functionality and Results

Role-Based Access Control (RBAC)

The system grants or denies access based on a user's role and permissions. Each role has specific actions they can perform on resources.

- **Doctor**: Can read, update, write, and delete.
- **Nurse**: Can read and update.
- **Admin**: Can manage, read, update, create, delete.
- **Pharmacist**: Can read, update, and write.
- **Receptionist**: Can read and create.
- **Lab Technician**: Can read and update.
- **Surgeon**: Can read, update, write, and delete.

Policy Evaluation Process

When a user attempts to access a resource, the system checks:

- **User Role**: The user's role is compared with the policy templates to see if they are authorized for the requested action.
- **Resource Type**: Each resource has associated policies that are evaluated based on the action requested (read, update, etc.).
- **User Attributes**: The system checks additional attributes such as the active shift status and whether the user is on a valid device and location.

Break-Glass Mechanism

In cases where users need access to resources they are not authorized for (emergency cases), the break-glass mechanism is invoked. The user is prompted for confirmation to bypass policy restrictions temporarily, and the event is logged for auditing.

Zero-Trust Verification

Before granting access, the system performs zero-trust checks:

1. **Active Shift Status**: Ensures the user is currently on an active shift.
2. **Device Authentication**: Verifies that the access request comes from an authorized device.
3. **Location Check**: Verifies the user is in an authorized location (e.g., hospital_1).

If any of these checks fail, access is denied, and the reason is logged.

Testing Scenarios and Results

1. Doctor Accessing Emergency EMR

- **User**: Dr. John (Role: Doctor)
- **Resource**: Emergency EMR (ID: 101)
- **Action**: Update
- **Result**: **Access Granted**
    - **Reason**: Doctors are authorized to read, update, write, and delete emergency EMR data. No issues with role-based access.
    - **Zero-Trust Check**: Passed (Device: device123, Location: hospital_1, Active Shift: Yes).

Log Output:

LOG: {'user': 'Dr. John', 'resource_id': 101, 'action': 'update', 'decision': 'PERMIT', 'reason': 'Policy evaluation passed', 'timestamp': '2025-02-01T12:15:00'}

2. Nurse Accessing Emergency EMR

- **User**: Nurse Jane (Role: Nurse)
- **Resource**: Emergency EMR (ID: 101)
- **Action**: Update
- **Result**: **Access Granted**
    - **Reason**: Nurses are authorized to read and update emergency EMR data, so access is allowed.
    - **Zero-Trust Check**: Passed (Device: device123, Location: hospital_1, Active Shift: Yes).

Log Output:

LOG: {'user': 'Nurse Jane', 'resource_id': 101, 'action': 'update', 'decision': 'PERMIT', 'reason': 'Policy evaluation passed', 'timestamp': '2025-02-01T12:20:00'}

3. Pharmacist Accessing Pharmacy Data

- **User**: Pharmacist Sam (Role: Pharmacist)
- **Resource**: Pharmacy Data (ID: 103)
- **Action**: Update
- **Result**: **Access Granted**
    - **Reason**: Pharmacists are authorized to update pharmacy data.
    - **Zero-Trust Check**: Passed (Device: device123, Location: hospital_1, Active Shift: Yes).

Log Output:

LOG: {'user': 'Pharmacist Sam', 'resource_id': 103, 'action': 'update', 'decision': 'PERMIT', 'reason': 'Policy evaluation passed', 'timestamp': '2025-02-01T12:25:00'}

4. Break-Glass Mechanism Triggered

- **User**: Nurse Jane (Role: Nurse)
- **Resource**: Emergency EMR (ID: 101)
- **Action**: Update
- **Break-Glass Triggered**: Yes
- **Result**: **Access Granted**
    - **Reason**: The nurse invoked the break-glass mechanism to access the resource. Access was granted for emergency purposes, and the event was logged.
    - **Log Output**:

LOG: {'user': 'Nurse Jane', 'resource_id': 101, 'action': 'update', 'decision': 'PERMIT', 'reason': 'Break Glass invoked by admin', 'timestamp': '2025-02-01T12:30:00'}

5. Zero-Trust Verification Failure

- **User**: Dr. John (Role: Doctor)
- **Resource**: Emergency EMR (ID: 101)
- **Action**: Update
- **Device**: unrecognized_device (fails device check)
- **Result**: **Access Denied**
    - **Reason**: The device ID does not match the expected device (device123).
    - **Zero-Trust Check**: Failed (Device: unrecognized_device).

Log Output:

LOG: {'user': 'Dr. John', 'resource_id': 101, 'action': 'update', 'decision': 'DENY', 'reason': 'Unrecognized device', 'timestamp': '2025-02-01T12:35:00'}

Logs and Auditing

All access attempts, whether granted or denied, are logged into access_logs.txt for auditing and compliance purposes. The logs capture critical information such as:

- **User**: The user performing the action.
- **Resource ID**: The resource the user is trying to access.
- **Action**: The action the user wants to perform.
- **Decision**: Whether the access was granted or denied.
- **Reason**: The reason for granting or denying access.

Future Work and Improvements

1. **Complex Policy Rules**: The current policy system is relatively simple. Future work could include implementing more granular policies, such as time-based access or multi-condition policies (e.g., user's department, shift time, etc.).
2. **Scalability**: The system is designed for a small set of resources and users. As the system grows, further optimization is needed, particularly in policy evaluation and decision-making.
3. **Real-Time Monitoring**: Implementing real-time monitoring and alerting for break-glass events and policy violations would enhance the security of the system.

Conclusion

This project implements an advanced access control system with a robust policy evaluation engine, zero-trust security checks, and a break-glass mechanism for emergency access. The system ensures that access is granted based on predefined roles and policies, and it logs all events for auditing and compliance. The system provides a secure and efficient method for managing access to sensitive healthcare data.