Cloud-Based Storage System - Security Measures Documentation

Project Repository: [Cloud-Based Storage System on GitHub]

1. File Encryption

Technique: Server-Side Encryption with Amazon S3 (SSE-S3)

- **Description:** We have implemented **Server-Side Encryption (SSE-S3)** to protect all files stored in Amazon S3 by encrypting them with the **AES-256 encryption algorithm**.
- **Purpose:** This encryption ensures that data at rest in the cloud is protected from unauthorized access.
- **How it Works:**
  - Each file is encrypted automatically on the server-side using AES-256 before being stored in the S3 bucket.
  - When a user uploads a file, Amazon S3 manages the encryption and decryption processes behind the scenes, ensuring a secure file storage system.
- **Benefit:** Even if an unauthorized individual accesses the data, they won't be able to read the file's contents without the encryption key.

2. Data Transmission Security

Protocol: Transport Layer Security (TLS)

- **Description: TLS encryption** is utilized for securing data transmitted between the user's device and the server, protecting the system from **man-in-the-middle (MITM)** attacks.
- **Purpose:** To ensure that any sensitive data sent over the network (such as files or credentials) is encrypted during transmission.
- **How it Works:**
  - The client connects to the server over a TLS-secured channel, ensuring that all data exchanged between the client and the server is encrypted.
  - This encryption protects data from being intercepted or altered while being transmitted.
- **Benefit:** Protects the integrity and confidentiality of the data as it moves between the user's device and the cloud.

3. User Authentication

Mechanism: OAuth 2.0 with JSON Web Tokens (JWT)

- **Description:** User authentication is handled through **OAuth 2.0**, where **JWT tokens** are used to securely authenticate users before they can access the system's functionalities.
- **Purpose:** To control access and ensure that only authorized users can upload, download, or delete files from the cloud-based storage system.
- **How it Works:**
  - Upon successful login, a JWT token is issued to the user. This token is used for subsequent requests to authenticate the user's identity.
  - The token is signed using the **HS256** algorithm and contains user-specific information (e.g., username) in a secure, encoded format.
  - On each API request, the server verifies the token to ensure that the user is authenticated before granting access to any resources.
- **Benefit:** Provides a scalable and secure mechanism to handle user authentication and authorization, ensuring unauthorized users cannot access or manipulate stored data.

4. Access Control

Model: Role-Based Access Control (RBAC)

- **Description: Role-Based Access Control (RBAC)** is implemented to manage permissions and restrict access based on user roles.
- **Purpose:** To ensure that only users with the correct roles (e.g., Admin or User) have access to specific system functionalities, such as file deletion or sharing.
- **How it Works:**
  - Users are assigned specific roles when they authenticate.
  - Permissions are assigned to these roles, and based on their roles, users are granted or denied access to certain operations.
- **Benefit:** Enhances security by ensuring that users can only perform actions that they are explicitly authorized for, preventing accidental or malicious actions.