Cyber Security Project : Web-Based Facial Authentication Systems

Yosri Zayani
Nada Hermi
Firas Amara
Mohammad Hajaj

I. System Overview

The proposed Web-Based Facial Authentication System aims to provide a robust and secure method for user authentication using facial recognition technology. This document outlines the design, components, and data flow of the system to achieve seamless and reliable authentication.

II. Components

- User Interface (UI):

Framework : Built with HTML, CSS, and JavaScript frameworks like React or Angular.
Provides an intuitive interface for user interaction, including enrollment and authentication processes.
Handles user interaction, captures user webcam input, and transmits data securely to the backend server.

- Backend Server:

Processes user requests, interacts with the facial recognition engine and database. Languages like Python with Django or Flask are popular choices.

- Facial Recognition Engine:

This system utilizes a facial recognition engine, acting as the core for real-time face detection, recognition, and verification. You have two main options for this engine:
1. Cloud-Based Services:
Leverage pre-built solutions offered by cloud providers like Amazon Rekognition, Microsoft Azure Face API, or Google Cloud Vision API.
These services offer ease of use and scalability, ideal if seeking a quicker setup with minimal development effort.
2. Open-Source Libraries (OpenCV):

For more control and customization, using OpenCV, a popular open-source computer vision library.
OpenCV provides libraries and pre-trained models for facial detection, landmark identification, and feature extraction.
This approach requires more development expertise but offers greater flexibility and potentially lower ongoing costs compared to cloud-based services.

- Database Management System (DBMS):

Stores user information (hashed facial templates and access permissions) securely. Popular choices include MySQL, PostgreSQL, or cloud-based database services. MongoDB is also used for storing user data, including faceprints and authentication logs.

- Web Server:

Manages communication between the web application and the backend server (if not using a serverless architecture). Ooptions include Apache or Nginx.

- Authentication Logic:

Implements Euclidean distance matching algorithm for comparing faceprints during authentication.

III. Working Flow

- User Enrollment:

- Users create an account by providing a username and password.
- Users grant access to their webcam for facial image capture through the web application UI.
- The captured image is sent securely to the backend server.
- The backend server utilizes OpenCV for face detection within the image.
- Once a face is detected, OpenCV extracts key facial features.
- A unique faceprint (hashed representation of features) is generated from the extracted data.
- User credentials (hashed password) and the faceprint are securely stored in the database.

- Authentication Process:

- Users enter their username and password on the login page.
- Upon successful login attempt, users grant access to their webcam for facial verification.
- The captured webcam image/frames are sent securely to the backend server.
- The backend server again utilizes OpenCV to process the image:

- Detecting faces within the frame.
- Extracting key facial features from the detected face.
- Generating a faceprint from the extracted data.
- The newly generated faceprint is compared against the registered faceprints stored in the database.
- The backend server receives a verification result (success/fail) based on the comparison.

● Verification and Access:

- If the verification result is successful (matching faceprint found) and the user has the necessary access permissions (if applicable), the backend server grants access to the web application.
- If the verification fails (no matching faceprint found or access denied), the user receives an error message and login is denied.
- The authentication result is displayed on the web application UI.

IV. Roles

● End Users:

Access web-based application and undergo facial authentication.

● System Administrators:

Manage user registrations, database maintenance, and system configurations.

V. Functions and Steps:

Enrollment Function:
User uploads facial image, OpenCV detects faces, extracts features, generates faceprints, which are stored along with user data.

Authentication Function:
User initiates authentication, OpenCV captures facial image, generates faceprint, compares it with registered ones, and displays the authentication result on the UI.

VI. Exchanged Messages/Data:

- Frontend to Backend:

User registration data (username, password)
Captured webcam image/video frames during login attempts

- Backend to Facial Recognition Engine:

User registration data (hashed facial template) for enrollment

Captured image/video frames for verification

- Facial Recognition Engine to Backend:

Verification results (success/fail)

- Backend to Database:

User registration data (hashed facial template, access permissions)
Verification requests (to retrieve user information)

- Database to Backend:

User information (access permissions) upon successful verification

VII. Authentication Rules:

- Users must create an account with a username and password for initial access.
- Login requires a successful username/password combination followed by facial verification through the webcam.
- Secure hashing functions are used to protect user passwords and facial templates.

VIII. Software and Libraries:

- Frontend Development: HTML, CSS, JavaScript frameworks (React, Angular)
- Backend Development: Python (Django, Flask), Node.js (Express)
- Facial Recognition Engine: Cloud-based options (Amazon Rekognition, Microsoft Azure Face API, Google Cloud Vision API), Open-source libraries (OpenCV with pre-trained models)
- Database: MongoDB, MySQL, PostgreSQL, Cloud-based database services (Amazon RDS)
- Web Server : Apache, Nginx (if not using a serverless architecture)

IX. Security Considerations:

User passwords are stored using secure hashing functions to protect against breaches.
Facial data is stored as hashed faceprints, not images, for added security even in case of a data leak.
Secure communication channels are used for data transmission between components.

X. Conclusion

This secure web-based facial authentication system leverages OpenCV for a user-friendly and secure login experience. By carefully considering the advantages and limitations, you can determine if an OpenCV-based approach aligns with your project requirements and development resources