Develop a ML based solution to refine captcha

[1]Shrilakshmi Prasad, [2]Theertha M, [3]Yukta Prakash,[4] Preethi G,[5] Sadiya Tarannum

[1]Professor, [2]Student, [3]Student,[4]Student,[5]Student
[1]Departement Of Computer Science and Engineering,
[1]ATME College Of Engineering, Mysore, INDIA

Abstract: The project introduces an image-based authentication mechanism called Captcha- Auth. Instead of traditional text-based passwords, users register by uploading personal images. At login, the system presents a mixed grid of user-enrolled images along with random distractors. The user must correctly identify their own images to gain access. This approach combines the strengths of CAPTCHA and graphical passwords to resist brute-force, dictionary, and observation attacks, while improving usability. This report covers system requirements, design methodology, architecture, data flow diagrams, test strategy, security analysis, and future enhancements. The control mechanism of CAPTCHA schemes depends on challenge-based solutions which humans can solve but automated programs cannot. The development of advanced automated systems has started to replace traditional CAPTCHA systems. The research field lacks complete understanding of automatic security enhancement methods for existing CAPTCHAs. The research presents a solution to enhance video-based CAPTCHA security through machine learning model-driven improvements. The security enhancements protect video- based challenges from multiple object-detection frameworks while providing enhanced protection against advanced automated synthetic media attacks. The system uses video challenges that consist of multiple object and action recognition tasks which remain resistant to resolution-preserving perturbation methods based on a substantial public dataset. The system handles both background elements and object detection within individual frames.

INTRODUCTION

Driver In today's digital era, ensuring secure user authentication has become a critical aspect of protecting sensitive information and online services. Traditional authentication methods, such as passwords or OTPs, are often vulnerable to phishing, brute-force attacks, or credential theft. To address these challenges, our system introduces a CAPTCHA-based secure authentication framework that combines password-based login with a personalized image-based puzzle, enhancing both security and user experience.

The system allows users to register by providing basic credentials—name, email, and password—along with three images of their choice. During registration, these images are processed through a Python-based YOLO (You Only Look Once) model to extract unique labels, which are stored in the backend database alongside user credentials. These labels later serve as a reference for generating the image puzzle during login.

Upon logging in, users first authenticate using their email and password. Following this, they are required to solve a personalized image puzzle, which consists of the three registered images interspersed with six distractor images fetched from the Unsplash API based on the image labels. Users must select the images in the exact order as during registration. This dual-layer authentication. mechanism significantly mitigates unauthorized access, as attackers must correctly identify the images in the correct sequence to proceed. To prevent misuse, the system allows a maximum of three attempts; exceeding this limit results in account blocking.

Blocked accounts are handled through a secure unblock process involving email- based OTP verification. Users are also given the option to update their images, which triggers a fresh YOLO analysis and updates the stored labels, ensuring that the authentication remains robust against potential security breaches. The system logs every puzzle attempt and image change to maintain an audit trail and facilitate monitoring.

Additionally, the system incorporates a data observation algorithm that analyzes puzzle attempts and image changes over the past seven days. Based on this analysis, users receive automated recommendations to update their images, enhancing security proactively.

Developed using a modern tech stack, the frontend leverages React for dynamic user interactions, while the backend integrates Spring Boot and Python Flask to handle user authentication, image processing, and puzzle generation. MySQL is used for persistent storage of user data and logs, while the Unsplash API provides diverse distractor images for puzzle complexity. This CAPTCHA-based secure authentication system thus combines the strengths of machine learning, secure authentication practices, and user-centric design to provide a robust, reliable, and intuitive mechanism for safeguarding user account.

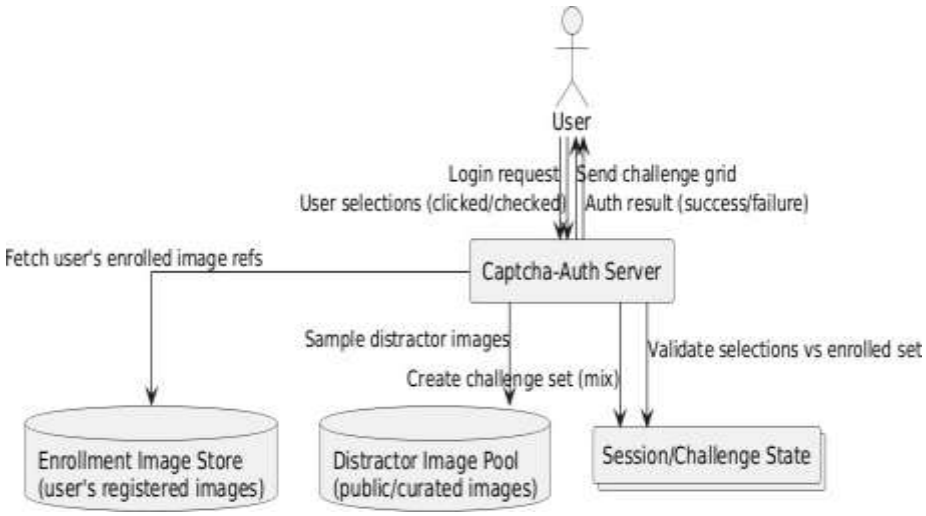
LITERATURE REVIEW

CAPTCHA, an acronym for Completely Automated Public Turing Test to Tell Computers and Humans Apart, is a computer test employed to differentiate humans from automated agents and is crucial for safeguarding systems against various cyber threats [3]. The most common cyber assault that CAPTCHA addresses is distributed denial of service (DDoS), in which attackers commandeer multiple machines to bombard a victim with overwhelming traffic. As a front-line defense against such sieges, CAPTCHA aids service providers in identifying attacking machines and blocking irregular traffic. Widely adopted for its low computational complexity, text-based CAPTCHA consists of alphanumeric strings distorted within noisy backgrounds, with enhancement

techniques including additional noise, overlay lines, and mixed-font styles. Yet CAPTCHAs must not use excessively similar character pairs, like '0' and 'O' or 'l' and 'I', since modern recognition algorithms can often decipher even noise-laden test samples. In addition to text-based challenges, image CAPTCHAs that require recognition of specific objects in provided pictures have gained popularity.

Besides serving a protective function, CAPTCHA systems facilitate the development of optical character recognition (OCR) algorithms by generating challenging samples that stress their capabilities and help build robustness to distortion. Given the increasing sophistication of malware techniques and their dedicated development to evade security systems, researching the vulnerabilities of contemporary CAPTCHA schemes and advancing the technology accordingly thus remains paramount. Acknowledging that CAPTCHA does not provide absolute security but merely slows the onslaught, effort should also be devoted to augmenting these systems with additional features, such as liveness checks, sensor-perfect requirements, and multi-factor signalling.

SYSTEM ARCHITECTURE



ARCHITECTURE OVERVIEW

1. Frontend Layer (ReactJS)

Provides a responsive and intuitive user interface for registration, login, puzzle verification, password reset, and image updates. Handles image uploads during registration and allows users to interact with the image puzzle seamlessly. Communicates with the backend via RESTful APIs to fetch puzzle images, verify selections, and manage OTP workflows. Implements client-side validations for input fields, image formats, and file sizes to ensure smoother user experience.

2. Backend Layer (Spring Boot + Flask)

Spring Boot Manages the main application logic including user authentication, session management, account blocking/unblocking, logging, and analytics. Flask (Python) Handles the ML component for image processing using the YOLO model. Each uploaded image is analyzed to generate unique labels for puzzle verification. Integrates with external APIs such as Unsplash to fetch distractor images dynamically based on labels. Ensures security by encrypting passwords, validating OTPs, and implementing account blocking logic after multiple failed attempts. Maintains logs for every puzzle attempt and image update, supporting both auditing and system analytics.

3. Database Layer (MYSQL)

Stores persistent user information, including credentials, uploaded images, YOLO-generated labels, and account status. Maintains detailed logs for puzzle attempts (attempts_logs) and image updates (image_changed_count_logs) to support monitoring and analytics. Supports relational queries for generating weekly analytics for the ML-based recommendation system.

ENTITY RELATIONAL MODEL

The database model comprises three main tables:

User: Stores user details: name, email, hashed password, image paths or blobs, YOLO-generated labels, account status (active/blocked), and registration metadata.

attempts_logs: Tracks every login puzzle attempt including timestamp, success/failure status, selected image sequence, and associated user ID.

image_changed_count_logs: Records each instance of image updates including timestamp, previous and new image references, and the user performing the update.

DATA FLOW SUMMARY

1.Registration: User uploads three images along with credentials. Images are sent to the Flask YOLO model to generate labels. Backend stores user credentials, images, and generated labels in the database

2. Login and Image Puzzle Verification: User logs in with email and password. Backend retrieves registered images and labels. Six distractor images are fetched from the Unsplash API based on the labels. User must select the registered images in the correct sequence to gain access. Every attempt is logged in attempts_logs.

3. Account Blocking: After three failed puzzle attempts, the account is blocked. Blocked accounts cannot access the image puzzle until the unblock process is completed.

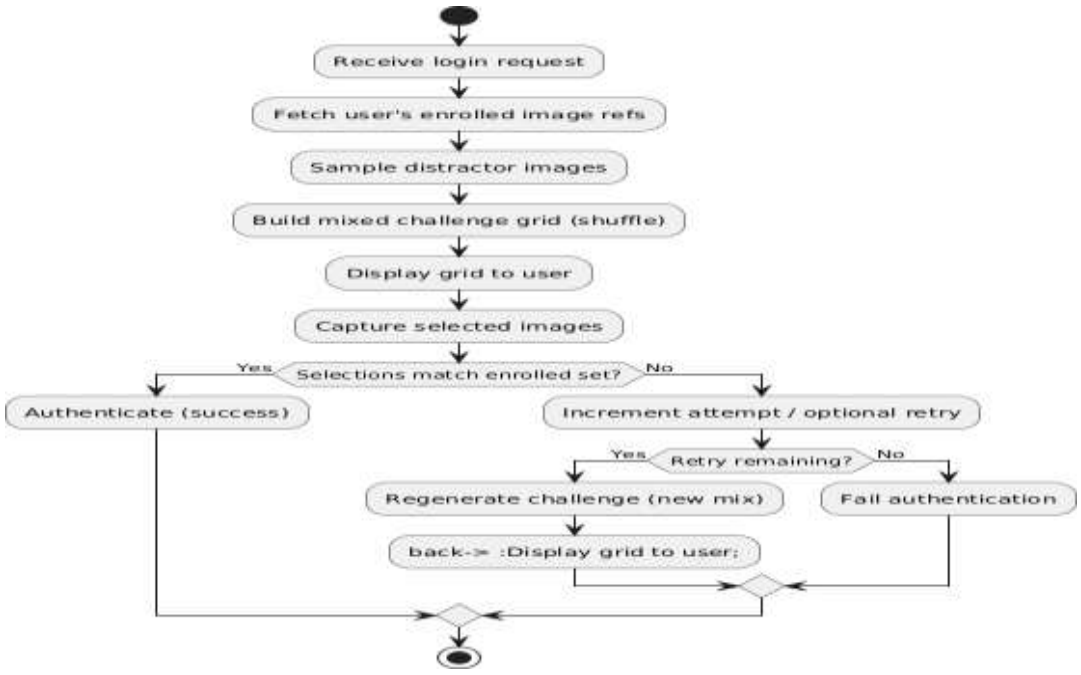
4.Unblocking: Users request an OTP to their registered email. Upon OTP verification, users can choose to update their images or retry the puzzle with the existing images. Image updates are logged inimage_changed_count_logs, and YOLO labels are updated in the database.

5. Analytics and Recommendation: A scheduled task collects the past seven days of puzzle attempts and image change logs. Based on this analysis, the system sends email recommendations prompting users to update their images if suspicious activity or repeated failures are observed.

6. Security and Logging: Every user action including login attempts, image uploads, OTP verification, and puzzle attempts is logged for auditing and system analytics. Sensitive data such as passwords and OTPs are securely stored and transmitted using encryption protocol.

IMPLEMENTATION

The project implements a multi-layered authentication system that combines traditional password login with a personalized image-based CAPTCHA for enhanced security. During registration, users upload three images, which are processed using a YOLOv8 model to extract unique object labels stored in the database. At login, after entering credentials, users must identify their registered images from a mixed set generated using the Unsplash API. Three failed attempts lead to account blocking, with recovery enabled through email-based OTP verification. The system also includes a data observation algorithm that monitors login behavior and image updates, suggesting timely security enhancements. Built using ReactJS, Spring Boot, Flask (for YOLO processing), and MySQL, the system ensures intelligent, adaptive, and user-friendly authentication.



TECHNOLOGY USED

Frontend - ReactJS
Backend - Spring Boot
ML Service - Python Flask with YOLO Database – MySQL
Used API to get distract images: Unsplash Image API

KEY MODULES

1.User Registration Module – Handles user image uploads and label extraction.
2.Login & Puzzle Verification – Manages password validation and image puzzle sequence.
3.OTP & Unblock Module – Controls account recovery and re-verification.
4.Logging Module – Records login attempts and image changes (Data Observation Algorithm)5.ML Module – Uses

YOLOv8 for object detection and labelling.

BACKEND INTEGRATION FLOW

User Controller (Spring Boot)
->/register endpoint
->callsFlaskAPIforlabels
->receiveslabelsJSON
->storesuserrecordinMySQL
Flask ML service runs YOLO Inference and returns label data

CONCLUSION

Machine Learning has a powerful capability to enhance security through analysis of data patterns. By analyzing millions of data patterns, CAPTCHAs security can be improved. However, despite enhanced security CAPTCHA is one of the tools that is losing its effectiveness. Existing CAPTCHAs are still vulnerable to ML- based recognition. This work optimizes one of the machines learning enhanced CAPTCHAs with some augmentations related to security. By using machine learning at the generation stage, CAPTCHAs are generated that are very easy for humans to solve but very hard for machine learning tools to recognize. Two kinds of security feature augmentations were appended to this already enhanced CAPTCHA tool. Three types of different datasets, and CAPTCHAs generated from the tool are used to evaluate these features. Augmenting liveness features will make more difficult for machine learning models to attack. There are many kinds of CAPTCHA generation tools with the help of machine learning. The combination of "easy for human" and "hard for machine" still remains a challenge.
This project successfully implements a multi-layered authentication system combining traditional password verification and image-based CAPTCHA validation It enhances user security by integrating machine learning and user behavioral tracking, ensuring adaptive and intelligent authentication.

REFERENCE

[1] Bin B. Zhu et al., "Captcha as Graphical Passwords—A New Security Primitive Based on Hard AI Transactions, 2024.
[2] R. Biddle et al., "Graphical Passwords: Learning from the First Twelve Years," ACM Computing Surveys, 2022.
[3] S. Wiedenbeck et al., "Pass Points: Design and Longitudinal Evaluation of a Graphical Password System," IJHCI, 2023.
[4] N. Dinh Trong, T. Ho Huong, and V. Truong Hoang, "New Cognitive Deep-Learning CAPTCHA," 2023. ncbi.nlm.nih.gov
[5] D. Hitaj, B. Hitaj, S. Jajodia, and L. V. Mancini, "Capture the Bot: Using Adversarial Examples to Improve CAPTCHA Robustness to Bot Attacks," 2020. [PDF]
[6] Z. Noury and M. Rezaei," Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment 2020. [PDF]
[7] X. Ling et al., "DEEPSEC: A uniform platform for security analysis of deep learning model," in Proc. IEEE Symp. Security Privacy, 2019, pp. 673–690.
[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. CVPR, 2016, pp. 770–778.