Password-less 2-Factor Authentication using RFID and facial recognition

Adam Board 2005335

CMP408

Introduction

Keeping an employee's account secure is important for an organisation as it reduces the chance of a data breach occurring. Using passwords puts a heavy responsibility on the user to choose and remember a strong password. Even if the password is strong, they can be stolen in a phishing attempt.

The purpose of this project was to implement password-less authentication with the combined use of RFID and facial recognition to ensure their account security is unique and personal. Removing the password from the process reduces the chance of an account being breached through brute-forcing or phishing attempts.

In this project, a prototype of password-less two-factor authentication was implemented using various hardware, software and cloud implementation. To consider this project a success, there were several objectives to achieve:

- A Kernel module to alert the Raspberry Pi of a button press
- A Kernel module to interact with LED and Buzzer through an interface
- A Python script on Raspberry Pi which can:
    - Scan RFID card to activate the camera.
    - Take a photo using camera and compare using AWS Rekognition.
    - If the photos match, provides generated code to unlock the webpage.
- An EC2 web server using Flask which:
    - Allows generated code to be inserted onto webpage.
    - Webpage to display personal details of user from S3 bucket.

Methodology

The project was split across two core sections, the Raspberry Pi and Amazon Web Services.

RASPBERRY PI: On the Raspberry Pi Zero W, physical hardware required setup. This includes, an LED, a buzzer, a button, a RFID scanner, and a camera module. Kernel modules were used to interface with the buzzer, LED and button.

The main Python script is contained on the Raspberry Pi. If the script detects the phrase "button_pressed" in the kernel log after the button is pressed, the RFID scanner activates for the first phase of the authentication process. If the correct card is used, the blue LED will flash three times before taking a photo with the camera. Afterwards, the photo is compared against an original photo stored in a S3 bucket using Amazon's Rekognition API. The "compare_faces" function will return the similarity between two different photos as a percentage. Based on Amazon's recommendations, the similarity percentage must be 99% or above to be accepted for authentication (Amazon, 2023). If the photo is accepted, a random code is generated for the user to insert into the web application hosted on an EC2 instance. If the authentication fails at any stage, the buzzer will beep three times and will require the user to redo the failed authentication. Using the Python library boto3, various AWS services are interacted with. Additionally, all credentials for access to AWS accounts are stored in external files for security purposes. An overview of the entire process on the Raspberry Pi can be seen in Figure 1.

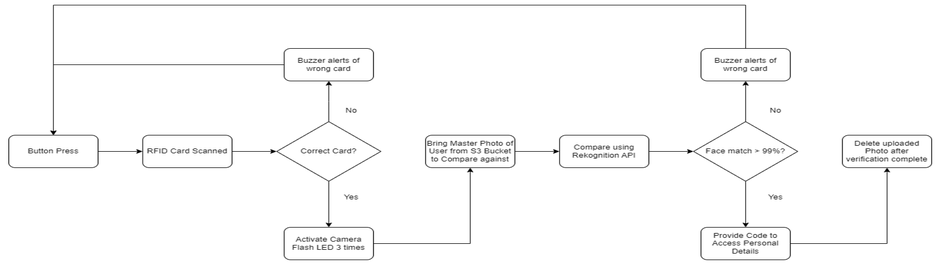

Figure 1 – Overview of Raspberry Pi Zero W Python Script

AWS: Two user accounts were created, one for the Raspberry Pi that is running the Python script, and one for the Flask web application. The Pi user account required permission to access Rekognition's "compare_faces" function, and the ability to add objects to the S3 bucket. The Flask web application user account required permission to list and get objects from the S3 bucket. The S3 bucket named "adamstorage" stored the comparison image and the personal information of the user. This bucket was made private to ensure only specific IAM users could access the objects.

A Flask application was setup on the EC2 Ubuntu instance to allow a user to access their personal information. Once the generated code is inserted into the main webpage, a presigned URL is generated which allows the Flask user account to access the objects inside of the private S3 bucket for a 5-minute time limit (CloudFlare, 2023). After grabbing the presigned URLs, the webpage displays the contents to the user. The S3 bucket is interacted with using the boto3 Python library

Project Highlights

This project achieved a successful implementation of a password-less authentication process using a combination of hardware, software, and the cloud. Key highlights of this project include:

- Use of Rekognition API in Python to compare the same face in two different photos to determine if it was the correct face.
- Implementing HTTPS using Caddy to both reverse proxy and automatically generate a certificate as proof of encrypted traffic
- Use of RFID scanner to detect different RFID cards and identify the correct card
- Using presigned URLs for the S3 bucket to allow temporary access to sensitive data instead of permanent access while keeping the bucket private.

Future Work

The project could be improved by implementing a lifecycle rule into the S3 bucket to delete generated codes after a period of time. This would reduce the total storage being used by unnecessary files and would aid in lowering the cost of running the authentication project in the future.

References

Amazon, 2023. Overview of face detection and face comparison. [Online] Available at: https://docs.aws.amazon.com/rekognition/latest/dg/face-feature-differences.html [Accessed 25 November 2023].

CloudFlare, 2023. Presigned URLs. [Online] Available at: https://developers.cloudflare.com/r2/api/s3/presigned-urls/ [Accessed 2 December 2023].