Real-Time Data Breach Alert System

Final Report

Name: K. Manichander
Program Name: Cybersecurity Assignment
Roll No.: 160123737043

GitHub Repository: https://github.com/Manichander123/Cybersecurity_

1 Project Overview

This project was completed as part of the Cybersecurity Assignment. The objective was to build a real-time data breach alert system that allows users to check if a password or email has been compromised in a known data breach. The system simulates a crucial security tool used to protect personal and corporate credentials.

The core functionality involves interacting with two reputable data breach APIs: Have I Been Pwned (HIBP) for password checking and XposedOrNot for email breach analysis. The project also includes a practical demonstration using the password cracking tool John the Ripper to show the risks of weak passwords.

2 Technologies & Tools Used

- **Backend Framework**: Python Flask
- **Data Breach APIs**: Have I Been Pwned (HIBP) and XposedOrNot
- **Password Cracking Tool**: John the Ripper (for the demo)
- **HTTP Library**: `requests`
- **Frontend**: HTML and CSS
- **Version Control**: Git & GitHub

3 System Architecture

The system is built as a web application using the Flask framework. The application's core architecture is as follows:

1. The user interacts with the system through a simple web interface developed with HTML and CSS.
2. When a user enters a password or email, a `POST` request is sent to the Flask backend.
3. The backend script (`app.py`) processes the request and sends a query to the respective third-party API.
4. For passwords, the application uses SHA-1 hashing and the k-anonymity protocol to query the HIBP API securely, without revealing the full password.
5. For emails, it queries the XposedOrNot API for breach analytics.
6. The application receives the response, parses the data, and displays a clear result to the user on the webpage.
7. A separate route is dedicated to a John the Ripper demonstration, showcasing how weak passwords are easily cracked from a list of hashes.

4 Key Features

- **Secure Password Check**: Implements the HIBP API's k-anonymity model to check passwords without exposing them in plaintext.
- **Email Breach Analysis**: Provides real-time information on whether an email address has been part of a data breach.
- **John the Ripper Demo**: A hands-on demonstration of password hashing and cracking using a real-world tool and wordlist.
- **User-Friendly Interface**: A clean, modern web interface with clear visual feedback for results.

5 Testing & Results

The application was tested extensively to ensure correct functionality and accurate results.

- **Password Check**:
  - When common passwords like '123456' or 'password' were entered, the application correctly reported that they were found in known breaches.
  - For a unique, strong password, the system confirmed that it was not found in the breach database.
- **Email Check**:
  - A known compromised email address (e.g., `abc123@gmail.com`) was correctly identified as being part of a breach.
  - A non-breached email address received a "Good news" response.
- **John the Ripper Demo**:
  - The tool successfully cracked the sample SHA-1 hashes using the provided wordlist, demonstrating the vulnerability of weak passwords.

The system successfully functions as a practical tool for basic data breach checking, providing reliable results and an educational demonstration of password security concepts.

6 Folder Structure

```
/DATA-BREACH-ALERT-SYSTEM/
 app.py
 index.html
 jtr-demo.pot
 jtr_demo.html
 requirements.txt
 sample_hashes.txt
 style.css
 wordlist.txt
```

7 Screenshots

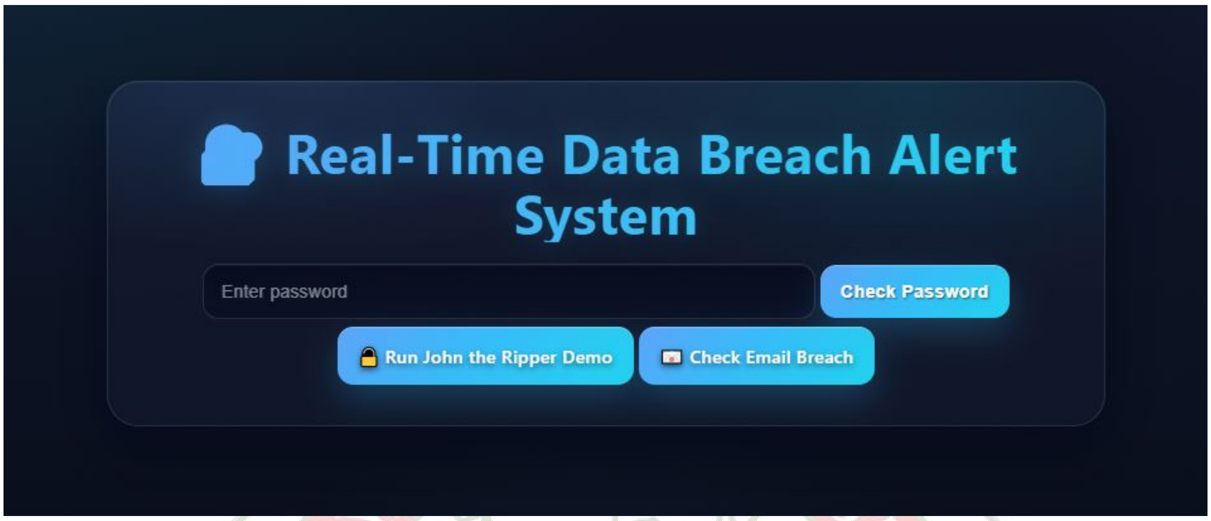
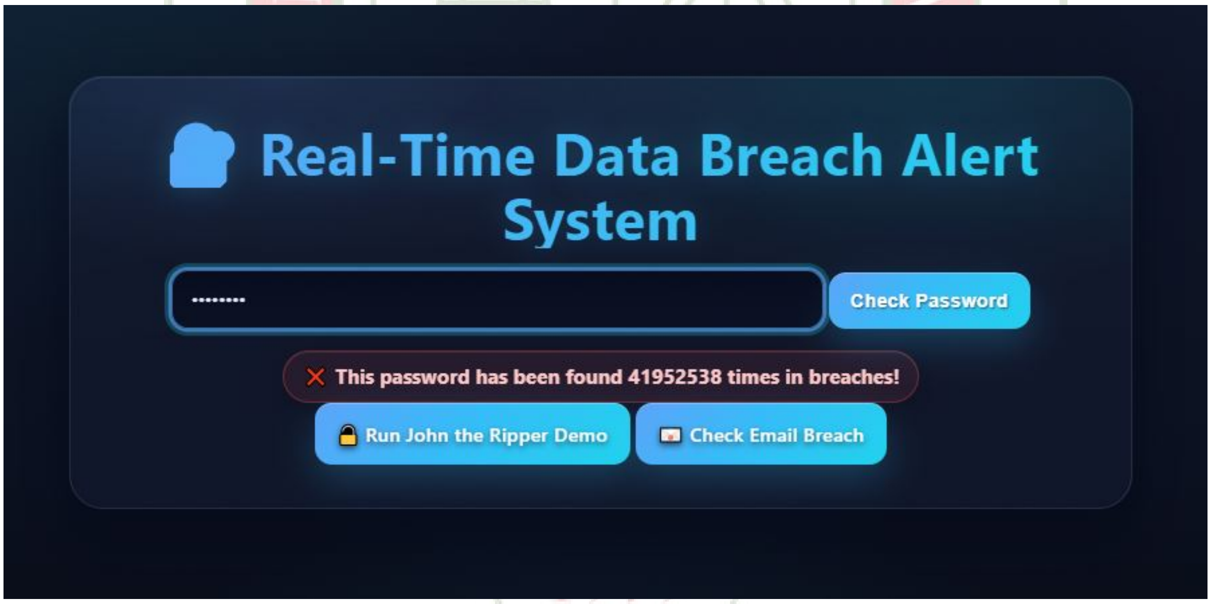
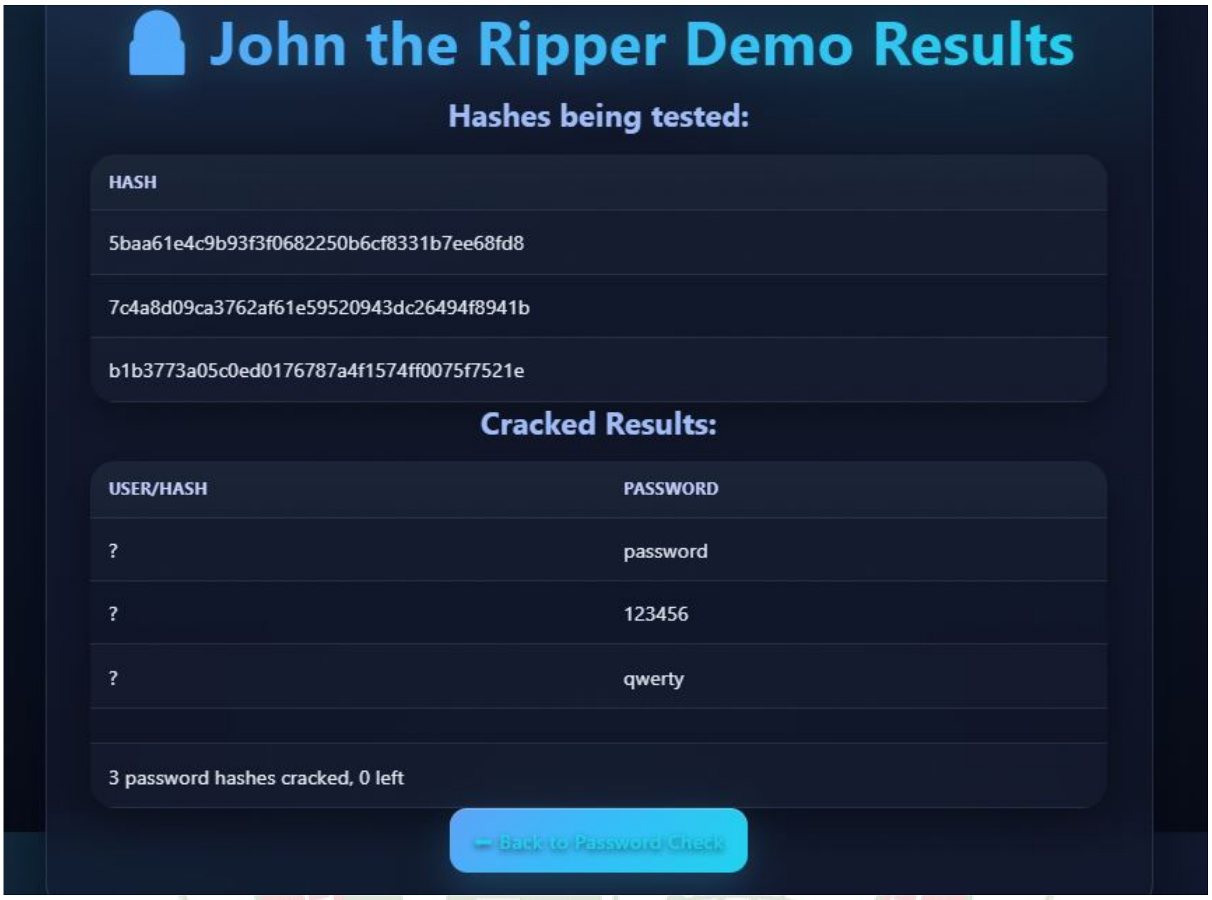


Figure 1: Home page showing the password check form.



This password has been found 41952538 times in breaches!

Figure 2: Result of a breached password check.



John the Ripper Demo Results

Hashes being tested:

HASH
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
7c4a8d09ca3762af61e59520943dc26494f8941b
b1b3773a05c0ed0176787a4f1574ff0075f7521e

Cracked Results:

USER/HASH	PASSWORD
?	password
?	123456
?	qwerty

3 password hashes cracked, 0 left

Figure 3: John the Ripper demo page showing cracked passwords.

8 Deliverables

- GitHub Repository with full project code and documentation.
- This Final Report document.

9 Learning Outcomes

- Gained practical experience in building a full-stack web application with Python Flask.
- Developed skills in integrating and utilizing external APIs for security-related tasks.
- Acquired a deeper understanding of password hashing, data breach analysis, and password cracking.
- Enhanced knowledge of secure coding practices and handling of sensitive data, including practical experience with a tool like John the Ripper.