Mini project report:

University / Faculty / Department:
university of Ferhat Abbas , computer science departememnt

Course / Module Name:
Privacy and security

Mini-Project Title:
Implementation of a Multi-Factor Authentication (MFA) System

Student Name(s):
Azib Wissam

Group / Section:
L2 big data

Academic Year:
2025– 2026

Supervisor:
Dr.Hadi fairouz

Abstract:

User authentication is an essential part of most computer systems, as it helps protect sensitive data and prevent unauthorized access. Many systems still rely only on passwords, which are often weak and can be easily compromised through attacks such as phishing or brute force. The main objective of this mini-project is to improve authentication security by implementing a Multi-Factor Authentication (MFA) system.

In this project, a simple authentication system was developed using Python. In addition to the traditional username and password, a second authentication factor based on a One-Time Password (OTP) was added. The system uses a graphical user interface (GUI) to make user interaction clearer and easier to understand. Several test scenarios were performed, such as entering an incorrect password or using an expired OTP, in order to evaluate the effectiveness of the proposed solution.

The obtained results show that adding a second authentication factor significantly increases system security. Even when the password is correct, access is denied if the OTP is invalid or expired. This project demonstrates that MFA is a practical and effective way to strengthen user authentication.

1/Introduction

Authentication is a key element in securing computer systems and applications. In many cases, access control is based only on a username and password. However, this method is no longer sufficient, as passwords can be guessed, stolen, or reused across different platforms. As a result, many security incidents are caused by weak or compromised credentials.

The motivation behind this project is to explore a stronger authentication mechanism that can reduce these risks. Multi-Factor Authentication (MFA) is widely used today to improve security by requiring users to provide more than one type of authentication information. By combining different factors, MFA makes it much harder for attackers to gain unauthorized access.

The objective of this mini-project is to design and implement a simple MFA system that combines a password with a One-Time Password (OTP). The project focuses on understanding the authentication process, implementing basic security mechanisms, and analyzing the benefits of MFA compared to single-factor authentication. This report is organized into several sections, starting with background concepts, followed by implementation details, results, and analysis.

2/Background and Related Concepts

Multi-Factor Authentication is based on the principle of combining independent authentication factors to increase security. Authentication factors are generally classified into three categories: something the user knows (password or PIN), something the user has (phone, token, or OTP), and something the user is (biometric data).

One-Time Passwords (OTPs) are temporary codes generated for a single authentication session. They are widely used as a second factor because they are time-limited and cannot be reused, which protects against replay attacks. OTPs can be delivered via SMS, email, or authenticator applications.

Password hashing is another essential concept used in secure authentication systems. Instead of storing passwords in plain text, cryptographic hash functions are used to store a fixed-length representation of the password, reducing the risk of credential exposure in case of data leakage.

3/Tools, Environment, and Technical Choices

The system was implemented using the Python programming language due to its simplicity, readability, and extensive standard library support. The ipywidgets library was chosen to build the graphical user interface, as it is included by default with Python and it is supported by google collab.

The hashlib library was used to implement secure password hashing using the SHA-256 algorithm. The random and time libraries were employed to generate OTPs and enforce expiration times.

The application was developed and tested on a standard personal computer running a desktop operating system with Python 3 installed. The choice of a desktop GUI application simplifies deployment and demonstration, making it suitable for educational purposes and reproducibility.

4/ System Design and Implementation

4.1 Overall System Architecture

The MFA system follows a modular architecture composed of three main components: user management, authentication logic, and the graphical user interface. The user management module handles registration and password storage. The authentication module manages login verification, OTP generation, and OTP validation. The GUI module provides interaction bottens for the user .

The workflow begins with user registration, followed by login using username and password. If the first factor is valid, the system generates an OTP and prompts the user to enter it. Access is granted only if both factors are successfully verified.

4.2 Implementation Details

Passwords are hashed using the SHA-256 algorithm before being stored in memory. During login, the entered password is hashed and compared with the stored hash. OTPs are generated as random six-digit numbers and stored along with an expiration timestamp. The OTP validity period is set to 30 seconds.

The graphical user interface includes separate screens for registration, login, and OTP verification. Error handling is implemented to manage incorrect credentials, invalid OTP formats, and expired OTPs.

4.3 Experimental Setup / Evaluation Methodology

The system was evaluated through multiple test scenarios simulating common attacks. These scenarios include attempting access with a compromised password, entering an incorrect OTP, and using an expired OTP. The system behavior was observed and recorded to assess its effectiveness.

5/ Results

The implemented system successfully authenticated legitimate users while blocking unauthorized access attempts. When incorrect passwords were provided, access was denied at the first authentication stage. When correct passwords but incorrect or expired OTPs were used, access was also denied.

Qualitative observations indicate that the system is responsive and user-friendly. The OTP expiration mechanism effectively prevents reuse of authentication codes. No successful unauthorized access was observed during testing.

6. Analysis and Discussion

The results obtained from the tests confirm that the implemented MFA system provides better security than a traditional password-based system. When only the password is required, an attacker who knows or guesses the password can easily gain access. With the addition of an OTP, this risk is significantly reduced.

During testing, the system behaved as expected. Access was denied when an incorrect or expired OTP was entered, even if the password was correct. This shows that the second authentication factor plays an important role in preventing unauthorized access and replay attacks.

However, the project also has some limitations. The OTP is displayed as a simulated message instead of being sent through a real email or SMS service. In addition, user data is stored only in memory, which means it is lost when the program is closed. Despite these limitations, the project successfully demonstrates the core principles and advantages of Multi-Factor Authentication.

7. Conclusion and Future Work

This mini-project successfully demonstrates the implementation of a Multi-Factor Authentication system using Python and a graphical user interface. The main objective of strengthening user authentication was achieved by combining a password with a One-Time Password as a second factor.

Through this project, practical experience was gained in implementing authentication mechanisms, handling password security, and managing OTP expiration. The results clearly show that MFA provides better protection than single-factor authentication and helps reduce common security risks.

For future work, the system could be improved by integrating real email or SMS services for OTP delivery, using time-based OTP applications such as Google Authenticator, and adding additional security features like account lockout and logging. The system could also be extended into a web-based application to explore client–server authentication scenarios.

Appendicies:

Source code: I implemented the MFA project in **Google Colab** using `ipywidgets` for interactive buttons and inputs. This version works fully in Colab, but **does not run in VS Code**, because `ipywidgets` requires a notebook environment to display GUI elements

..\Desktop\sp project\sc.py