Mini Project Report: Password Strength Checker

Project Details

Project Title: Password Strength Checker

Technology Used: Python, Regular Expressions (re module)

Project Type: Mini Project

Domain: Cybersecurity / Python Programming

Duration: 1 day

Status: Completed

1. Abstract

In an increasingly digitized world, passwords remain a primary method of securing access to online systems, accounts, and sensitive data. However, the effectiveness of a password hinges on its strength, which determines how resistant it is to brute-force attacks, dictionary attacks, and guesswork. This project presents a lightweight yet effective password strength checker built using Python's re module for regular expression processing. The tool evaluates passwords based on a combination of common industry security standards, providing both qualitative (weak, moderate, strong) and quantitative (percentage-based) feedback to guide users in creating robust passwords.

2. Introduction

Passwords are the most common authentication mechanism in digital systems, from web applications to enterprise systems and IoT devices. Weak or reused passwords are one of the leading causes of data breaches. Despite the growing awareness of cybersecurity threats, many users still choose passwords that are easy to remember—and easy to crack.

This project aims to promote security hygiene by providing users with immediate feedback on password strength. By integrating both visual and statistical analysis of password quality, users can make informed decisions and understand the vulnerabilities of their chosen credentials.

3. Objective

To build a tool that evaluates the strength of a user's password based on several security criteria and categorizes it as Strong, Moderate, or Weak.

4. Features

- Checks for:

  • Minimum password length (≥ 8)

  • Presence of digits

  • Presence of uppercase letters

  • Presence of lowercase letters

  • Presence of special characters

- Calculates a strength percentage out of 100%

- Returns clear, readable output for each criterion

- User-friendly interface via CLI (Command-Line Interface)

5. Learning Outcomes

- Gained proficiency in Python's re module

- Learned how to define and apply security criteria

- Understood logic design for real-world tools

- Practiced output formatting and feedback clarity

6. Technologies Used

Python 3.13 – The primary programming language

re module – Used to perform pattern matching via regular expressions

CLI / Terminal – User interface for interaction

7. Methodology and Algorithm

7.1 Criteria Used:

- Minimum Length (≥ 8 characters)

- Contains at Least One Digit (\d)

- Contains At Least One Uppercase Letter ([A-Z])

- Contains At Least One Lowercase Letter ([a-z])

- Contains At Least One Special Character ([!@#$%^&*(),.?":{}|<>])

7.2 Logic:

- Rule-Based Classification: Weak / Moderate / Strong based on how many criteria fail.

- Scoring Model: Each successful criterion adds 20% to a total score out of 100%.

7.3 Regular Expressions:

- re.search(r"\d", password) – checks for digits

- re.search(r"[A-Z]", password) – checks for uppercase letters

- re.search(r"[a-z]", password) – checks for lowercase letters

- re.search(r"[!@#$%^&*(),.?\":{}|<>]", password) – checks for special characters

8. Execution Flow

1. User is prompted to input a password.

2. The password is checked against all 5 rules.

3. A dictionary holds the boolean results of each rule.

4. A cumulative score and corresponding strength level is displayed.

5. The output gives a percentage and a pass/fail breakdown of all criteria.

9. Sample Output

Input:

Enter your password: Cyber@123

Output:

Password Strength: 100% - Strong

Criteria:

Length >= 8: yes

Has digit: yes

Has uppercase letter: yes

Has lowercase letter: yes

Has special character: yes

10. Applications and Use Cases

- Signup/Login forms for websites and mobile apps

- Corporate password policies as a validation step

- Offline utilities for password auditing

- Educational demos for teaching secure authentication practices

11. Limitations

- Doesn’t check for dictionary words or breached passwords

- Doesn’t evaluate password entropy

- No integration with password managers or hashing

- GUI not included (CLI only)

12. Future Scope

- Add entropy-based evaluation (Shannon entropy)

- Cross-check against leaked password databases

- Build a Tkinter or Flask-based GUI/web interface

- Add password generation feature

- Use ML models to estimate crackability

13. Conclusion

This project successfully demonstrates how a rule-based and percentage-scored password strength checker can be implemented in Python using regular expressions. It offers valuable feedback for users looking to improve the strength of their credentials and encourages security best practices. While basic, it sets a solid foundation for further development into a more comprehensive password security suite.