Deerwalk institute of training center

Sifal, Kathmandu, Nepal

Project:- To check the strength of password and make password generator if the strength is low.

Submitted By:

Pankaj Mahato

(Student of cyber security)

Deerwalk Institute of Technology

Sifal, Kathmandu

Nepal

Submitted To:

Apurva Nepal

(Teacher/Trainer of cyber security)

Deerwalk Institute of technology

Sifal, kathmandu, Nepal

Date:-2025-03-17

Signature of Prabin Thapliya

Training Manager

Signature of Apurva Nepal

Subject Teacher

A**cknowledgement**

I would like to express my sincere gratitude to my teacher, **Mr. Apurva Nepal**, for his invaluable guidance, encouragement, and support throughout this project. His expertise in cybersecurity and insightful feedback have greatly helped me in understanding the concepts of password security and implementing them effectively in Python.

I am also thankful to **Deerwalk Institute of Technology** for providing me with the opportunity and resources to work on this project. The knowledge and skills acquired during this study have been instrumental in enhancing my understanding of cybersecurity.

Lastly, I would like to thank my friends, and peers of my class for their continuous motivation and support, which have been essential in completing this project successfully.

Pankaj Mahato

Abstract

In today's digital era, ensuring the security of passwords is crucial for protecting sensitive information. This project focuses on developing a **password strength checker and generator** using Python. The password strength checker evaluates the security level of a given password based on various factors such as length, character diversity, and the presence of common patterns. If the password is found to be weak, the system generates a strong and secure password based on best security practices.

The implementation involves techniques such as **regular expressions, entropy calculations, and randomness functions** to assess and enhance password security. This project aims to raise awareness about password safety and encourage users to adopt stronger authentication practices. By integrating this tool into applications, individuals and organizations can significantly improve their cybersecurity posture.

This project is submitted as part of the academic requirements at **Deerwalk Institute of Technology** under the guidance of **Mr. Apurva Nepal**.

Keywords: Password Strength, Cybersecurity, Python, Strong Password Generation, Security Best Practices.

Table of Contents Page.No.

1. Project Introduction

In the modern digital world, passwords serve as the first line of defense in protecting user accounts and sensitive data from unauthorized access. However, weak and easily guessable passwords continue to be a major cybersecurity risk, making individuals and organizations vulnerable to cyber threats such as brute-force attacks, dictionary attacks, and credential stuffing. To mitigate these risks, it is essential to evaluate password strength and encourage the use of strong passwords.

1.1. Executive Summary:

1.1.1. Purpose:

The purpose of this project is to enhance password security by evaluating the strength of user-provided passwords and generating strong passwords when necessary. Weak passwords are a major security risk, and this project helps users create more secure credentials by providing real-time feedback and suggestions.

1.1.2. Expected Outcomes:

- **Password Strength Analysis** – The program will check if a password meets security criteria such as length, uppercase/lowercase letters, numbers, and special characters.
- **Weak Password Detection** – If a password is weak, the user will be alerted with suggestions for improvement.
- **Secure Password Generation** – If a password is too weak, the system will generate a strong, randomized password that meets best security practices.
- **User-Friendly Experience** – The project will provide an easy-to-use interface with clear feedback and recommendations.

1.2. Project Objective:

Goals:

- **Enhance Password Security** – Help users create strong passwords to protect their accounts from cyber threats.
- **Educate Users on Strong Password Practices** – Provide feedback on password strength and suggest improvements.
- **Automate Secure Password Generation** – Generate strong passwords for users who enter week ones.
- **Improve Usability** – Offer a simple, user-friendly interface for password assessment.

Objectives:

- **Detect Weak /strong Passwords** – Identify passwords that are easy to guess or commonly used and make it easily not guessable.

- **Provide Feedback and Suggestions** – Inform users about their password's weaknesses and how to strengthen them.

- **Generate Strong Passwords** – Automatically create secure passwords with a mix of uppercase, lowercase, numbers, and special characters.

- **Ensure Security Best Practices** – Prevent common vulnerabilities by discouraging predictable password patterns.

1.3 Project Description:

1.3.1. Password Strength Checker:

The program evaluates passwords based on key security factors:

- **Length Requirement:**
  - Minimum **8 characters** for a password to be considered strong.
  - Encourages longer passwords (12-16+ characters) for better security.

- **Character Diversity:**
  - **Uppercase letters** (A-Z)
  - **Lowercase letters** (a-z)
  - **Numbers** (0-9)
  - **Special characters** (!@#$%^&* etc.)

- **Pattern Detection:**
  - Identifies **common words** (e.g., "password", "123456", "qwerty").
  - Detects **repeated or sequential characters** (e.g., "aaaaaa" or "abc123").

- **Entropy Calculation (Optional Advanced Feature):**
  - Measures randomness in the password to determine its strength

1.3.2. Password Strength Levels:-

The program classifies passwords into three categories:

- **Strong Password (Score: 5/5):**

  Meets all security criteria (length, uppercase, lowercase, number, special character).

- **Moderate Password (Score: 3/5):**

  Meets some but not all criteria; suggests improvements.

- **Weak Password (Score: 2/5):**

  Easy to guess, lacks diversity; system generates a stronger password.

1.3.3. Libraries Used:

- **re –** For pattern matching and password validation.
- **random** – For generating strong passwords.
- **string** – Provides character sets for password generation.
- **getpass** – Ensures secure password entry.
- **Tkinter**- GUI library.

1.3.4. Tkinter (Built-in GUI Library):

 Why Use It?

✓ Comes pre-installed with Python.
✓ Easy to use for simple GUI applications.
✓ Lightweight and does not require additional dependencies.

 Best For: Small applications with basic GUI needs.

1.4. Project scope:-

1.4.1. Scope Inclusions (What is Included?):

- **Password Strength Analysis** – Evaluates passwords based on:

  - Length (minimum 8-16+ characters for security).
  - Character diversity (uppercase, lowercase, numbers, special characters).
  - Pattern detection (common words, repeated sequences).

- **Password Classification** – Categorizes passwords as:

  - Weak – Lacks security elements.
  - Moderate – Partially secure, needs improvement.
  - Strong – Meets all security standards.

- **Security Feedback & Suggestions** – Provides tips for improving weak passwords.
- **Secure Password Generator** – Generates random, strong passwords if input password is weak.
- **Randomization for Security** – Ensures unpredictability in generated passwords.
- **User-Friendly Console Interface** – Simple command-line-based user interaction.
- **Graphical User Interface (GUI) (In the Basic Version)** – Initial implementation is console-based**.**

1.4.2. Scope Exclusions (What is NOT Included?):

❌ **Password Storage & Management** – The system does not save passwords for security reasons.

❌ **Cloud or Database Integration** – No online data storage or cloud services.

❌ **Encryption & Secure Vaults** – The project does not function as a password manager.

❌ **Integration with External Systems** – No direct connection to websites or authentication services.

❌ **AI-Based Security Analysis** – No machine learning or advanced AI techniques for password.

2. Related works:

The topic of **password security** has been widely explored in cybersecurity research, software development, and security policies. Several existing tools and methodologies help assess password strength and generate secure passwords. This section highlights relevant studies, tools, and approaches related to **password strength checking and password generation**.

2.1 Existing Password Strength Checkers:

Several password strength evaluation tools are available, each using different security parameters:

2.1.1. *NIST Password Guidelines:*

- The **National Institute of Standards and Technology (NIST)** provides guidelines for password security.
- It recommends using **long, complex passwords** and avoiding dictionary words or repetitive patterns.
- Modern password strength checkers follow **NIST SP 800-63B** for assessing password security.

2.1.2. Zxcvbn by Dropbox:

- A **real-time password strength estimator** developed by Dropbox.
- It analyzes common words, keyboard patterns, and frequently used passwords.
- Uses a **scoring mechanism (0 to 4)** to classify passwords.

2.1.3. Online Password Checkers:

- Websites like **How Secure Is My Password?** use algorithms to estimate how long it would take to crack a password.
- Browsers like **Google Chrome** and **Firefox** have built-in password strength indicators.

2.2. Existing Password Generators:

Password generators create **secure and random passwords** based on best security practices. Some well-known tools include:

2.2.1. LastPass & 1-Password Generators:

- These **password managers** provide built-in generators to create **random, strong passwords**.
- They follow security policies such as **minimum length, special character inclusion, and randomness**.

2.2.2. Diceware Method:

- Uses **random words** to create an easy-to-remember yet secure password.
- Example: correct-horse-battery-staple (random words chosen from a word list).

2.2.3. Operating System Built-in Generators:

- Linux's pwgen command generates random passwords.
- Windows and macOS have **password suggestion features** integrated into browsers and authentication systems.

2.3. Research & Algorithms for Password Strength Checking:

Several **academic and industry studies** have explored methods to measure password security:

2.3.1. Entropy-Based Strength Evaluation:

- Measures password **unpredictability** based on Shannon entropy.
- **Higher entropy** means a stronger password.

2.3.2. Pattern Recognition & Dictionary Attacks:

- Detects **common words, keyboard patterns, and predictable sequences**.
- Example: "12345678" and "password123" are weak due to high predictability.

3. System Analysis:

The **strength of the system** lies in its ability to analyze password security effectively and generate **strong, random passwords** when needed. Below is an in-depth analysis of its strengths:

3.1. Strengths:

3.1.1. Strong Password Generation:

- **Randomized Output** – The generated passwords are completely random and not based on patterns.
- **Customizable Complexity** – Users can define password length and character mix**.**
- **High Entropy** – Generated passwords have strong resistance to brute-force attacks**.**

3.1.2. Performance & Efficiency:

- **Fast Processing** – Password evaluation and generation happen in milliseconds**.**
- **Lightweight & Efficient** – Does not require large computing resources or external databases.
- **Scalable** – Can be expanded with additional features like AI-powered security analysis or database integration**.**

3.1.3. Usability & User Experience:

- **Simple & Intuitive** – Uses a console-based interface (easy to understand and interact with).
- **User Guidance** – Provides feedback & improvement tips for weak passwords.
- **Portable** – Can run on any system with Python installed (Windows, Linux, macOS).

3.1.4. Flexibility & Extensibility:

Easily Customizable – Can be adapted for different security policies (e.g., corporate password policies).

Potential for GUI – Future versions can integrate Tkinter, PyQt**, or** Kivy for a graphical interface.

Integration Ready – Can be extended for use in web applications, authentication systems, or password managers.

3.2. Weaknesses of the System:

3.2.1. Lack of Breached Password Detection:

- The system does not check if a password has been leaked in data breaches**.**
- Attackers can easily crack commonly used passwords, even if they seem strong.
- **Solution:** Future versions could integrate **"Have I Been Pwned" API** to check against known breached passwords.

3.2.2. No Advanced Security Analysis (Entropy & AI-Based Checking):

- The system only evaluates **basic password parameters** (length, character variety, common patterns).
- It does not calculate **password entropy** (randomness measurement).
- It does not use **machine learning** to detect password weaknesses.
- **Solution:** Add **entropy calculation** or AI-based analysis for better security evaluation.

3.2.3. Limited User Guidance:

- Users receive **basic feedback** on password strength but no **detailed guidance** on choosing a secure password.
- **Solution:** Provide **step-by-step recommendations** (e.g., suggest using passphrases instead of random strings).

3.2.4. Weak Password Generation Algorithm (Lack of Cryptographic Strength):

- The system relies on Python's random module, which is **not cryptographically secure**.
- Attackers could predict generated passwords using randomization weaknesses.
- **Solution:** Use secrets module instead of random for truly **unpredictable** password generation.

3.2.5. No Multi-Language Support:

- The system currently supports **only English**, which limits accessibility for non-English users.
- **Solution:** Implement **multi-language support** for broader usability.

3.2.6. No Multi-Platform Support:

- The program is designed for Python-based execution, making it **less accessible for non-programmers**.
- **Solution:** Convert the program into a **web-based tool or mobile app** for better usability.

3.2.7. No Two-Factor Authentication (2FA) Integration:

- The system focuses on password strength but does not recommend or enforce **2FA** for better security.
- **Solution:** Provide suggestions for **multi-factor authentication (MFA)** to improve user security.

3.2.8. Future Improvements & Solutions:-

- using **cryptographically secure** password generation (secrets module).
- Integrating **breach detection APIs** (e.g., Have I Been Pwned).
- Improving **password feedback** with detailed security recommendations.
- Providing **password management** with encrypted storage.
- Adding **entropy calculation & AI-based security analysis**.

4. Technical Analysis:-

4.1. System Architecture:

4.1.1. Modules & Components:

The system consists of the following core modules:

- Password Strength Checker:

  Takes a user-input password and evaluates its strength based on:

  - Length (minimum 8–16+ characters)
  - Character diversity (uppercase, lowercase, numbers, special characters)
  - Common password detection (checks for weak/common passwords)
  - Password entropy calculation (randomness measure)
  - Dictionary attack prevention (detects common words)

- Outputs password strength as **Weak, Moderate, or Strong**.

- Password Strength Feedback

  - If a password is weak, the system provides:
    ☑ A security score based on password complexity.
    ☑ Recommendations to improve the password (e.g., "Use a mix of uppercase, lowercase, numbers, and special characters").

4.2. Technology Stack:

4.2.1. Programming Language: Python:

- Python is used due to its **simplicity, security, and wide range of libraries** for password management.

4.3. Password Strength Algorithm:

4.3.1. Steps to Evaluate Password Strength:

- **Check length** → If < 8 characters, mark as Weak.
- **Check character variety** → Does it contain uppercase, lowercase, numbers, and symbols?
- **Detect common passwords** → Compare against a list of known weak passwords.
- **Check dictionary words** → Avoid single dictionary words (e.g., "apple", "admin").
- **Entropy calculation** → Measure randomness based on Shannon entropy formula.
- **Final classification** → Output "Weak", "Moderate", or "Strong".

4.3.2. Password Strength Classification Example:

Criteria	Weak	Moderate	Strong
Length	< 8 chars	8-12 chars	12+ chars
Character diversity	Low (letters only)	Moderate (letters + numbers)	High (letters, numbers, symbols)
Common words	Yes	No	No
Entropy	< 40 bits	40-60 bits	60+ bits

4.4. Secure Password Generation:

Algorithm for Generating Secure Passwords:

- **Set password length** (default: 12-16 characters).
- **Use secrets.choice()** to select random characters.
- **Ensure mix of uppercase, lowercase, numbers, and symbols**.
- **Shuffle characters to prevent patterns**.
- **Return password to user**.

Example Generated Passwords:

☑ aP$8x%Km!7rY (Strong)
☑ Qz@9Tj4&uM$V (Strong)
❌ 12345678 (Weak – contains only numbers)

System Workflow:

Step 1: User inputs a password.
Step 2: System analyzes password strength.

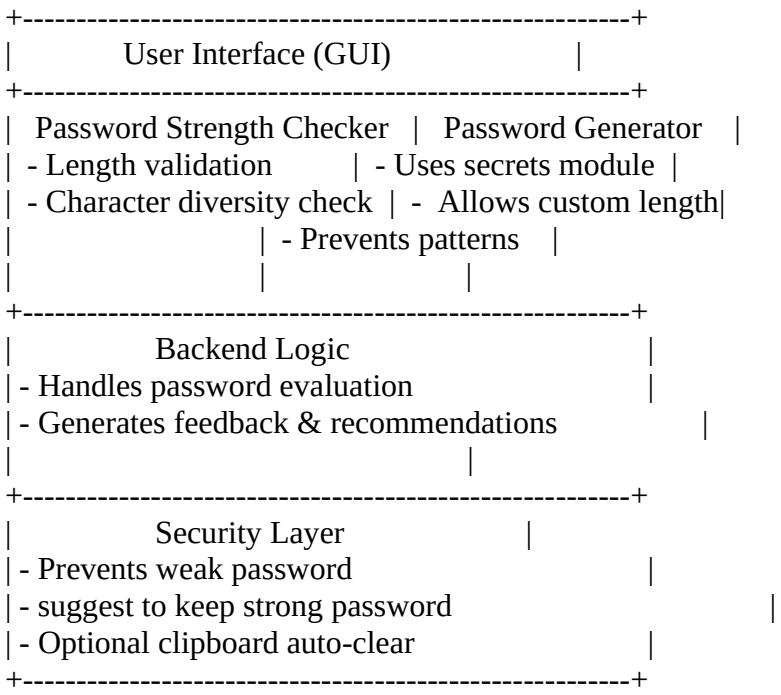
- ☑ If **Strong** → Display success message.
- ❌ If **Weak** → Show warning and improvement suggestions.

5. System Design:

5.1. System Architecture:

The system is structured into multiple **layers and modules**, ensuring modularity and scalability.

- **Architecture Diagram (High-Level Overview):**



5.2. Components & Modules:

Input Handling Module:

- ✓ Accepts user-input passwords via **GUI**.
- ✓ Ensures password is **not empty** before evaluation.
- ✓ Supports **user-defined password length** (for generation).

Password Strength Checker Module:

- ✓ **Validates password length** (Minimum 8 characters).
- ✓ **Checks character variety** (uppercase, lowercase, numbers, symbols).
- ✓ **Compares against common weak passwords** (e.g., "password123", "qwerty").
- ✓ **Detects dictionary words** to prevent easy guesses.
- ✓ **Calculates password entropy** to measure randomness.
- ✓ Classifies passwords as **Weak, Moderate, or Strong**.

5.3. Technology Stack:

Component	Technology
Programming Language	Python
GUI Implementation	Tkinter
Secure Password Generation	secrets module
Regex & Pattern Matching	re module

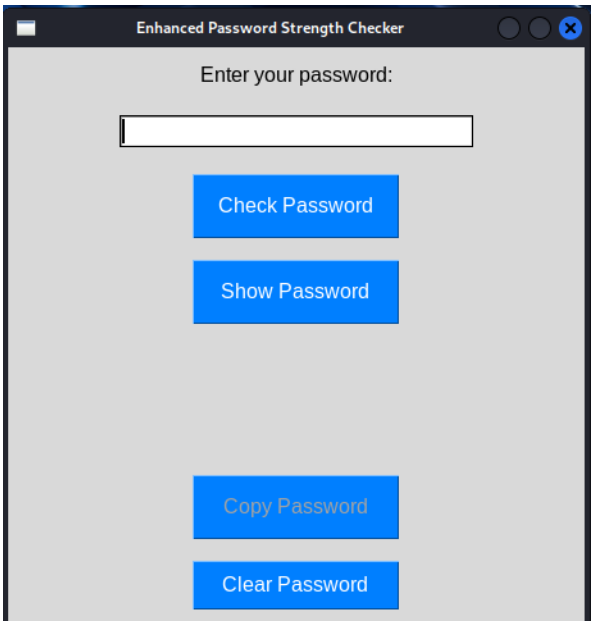
5.4. Security Considerations:

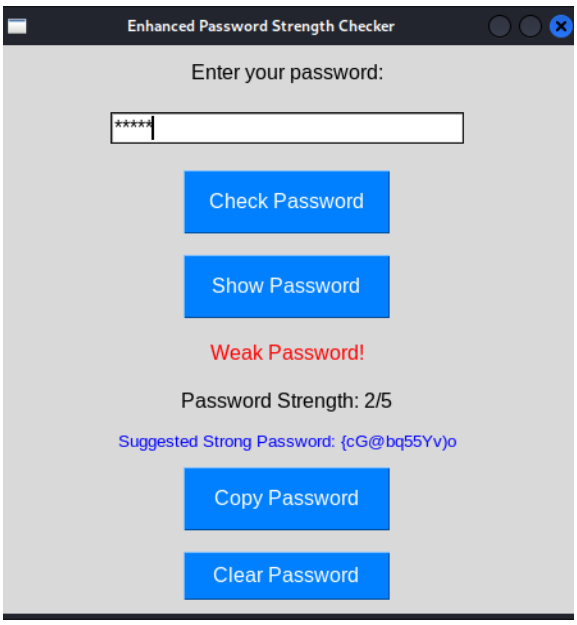
- ✓ **No password storage** to prevent data leaks.
- ✓ Uses **cryptographic secure random generator** (secrets).
- ✓ Protects against **brute force & dictionary attacks**.
- ✓ Future enhancement: Integration with **breached password databases**.

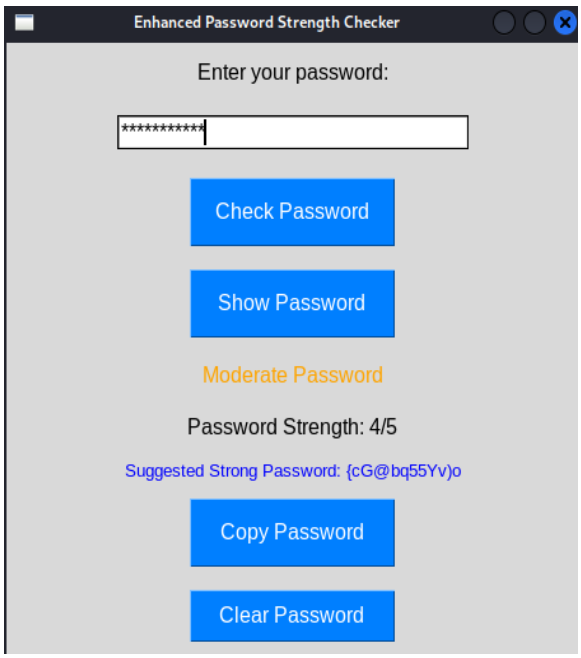
6. Pseudocode:

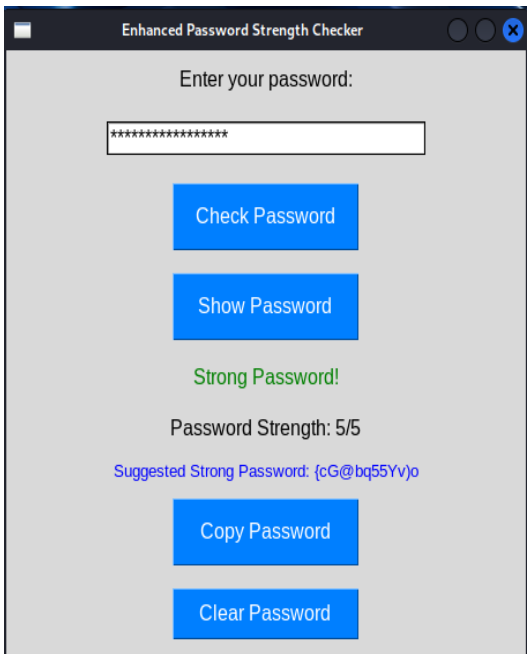
```
#library included
import tkinter as tk
from tkinter import ttk
import re
import random
import string

# Function to check password strength
check_password_strength(password):
    Initialize strength_level = 0
    If password length >= 8, increase strength_level
    If password has uppercase, increase strength_level
    If password has lowercase, increase strength_level
    If password has digits, increase strength_level
    If password has special characters, increase strength_level
    Return strength_level

# Function to suggest a strong password
suggest_strong_password():
    Return a random 12-character password (letters, digits, special characters)

# Function to handle password check and display results
check_password():
    Get password input
    Check password strength
    If strength == 5, display "Strong Password"
    If strength == 3 or 4, display "Moderate Password"
    If strength < 3, display "Weak Password" and suggest a strong password

# Function to clear password entry
clear_password():
    Clear password input and reset labels

# Function to toggle password visibility
toggle_password():
    Toggle between showing and hiding the password

# Function to copy suggested password to clipboard
copy_to_clipboard():
    Copy suggested password to clipboard

# Button hover animations
on_hover():
    Change button color on hover
```

7. Output:


Enhanced Password Strength Checker
Enter your password:
Check Password
Show Password
Copy Password
Clear Password


Enhanced Password Strength Checker
Enter your password:
*****
Check Password
Show Password
Weak Password!
Password Strength: 2/5
Suggested Strong Password: {cG@bq55Yv)o
Copy Password
Clear Password


Enhanced Password Strength Checker
Enter your password:
****
Check Password
Show Password
Moderate Password
Password Strength: 4/5
Suggested Strong Password: {cG@bq55Yv)o
Copy Password
Clear Password


Enhanced Password Strength Checker
Enter your password:
****
Check Password
Show Password
Strong Password!
Password Strength: 5/5
Suggested Strong Password: {cG@bq55Yv)o
Copy Password
Clear Password

8. Conclusion:

The **Password Strength Checker & Generator** project successfully addresses the need for secure password management by evaluating password strength and generating strong passwords when necessary. The system ensures users create **strong, complex, and secure passwords**, reducing the risk of cyber threats such as brute-force attacks, dictionary attacks, and credential stuffing.

This project contributes to cybersecurity awareness by educating users on password best practices and encouraging them to adopt **stronger authentication measures**. Future improvements, such as **integration with online breach-checking services (e.g., Have I Been Pwned API)** and **password manager support**, could further enhance its effectiveness.

Overall, this project is a **valuable tool for improving personal and organizational security**, making it easier for users to create and maintain strong passwords while minimizing the risks associated with weak credentials.

Refrence:

- https://chat.deepseek.com/
- https://chatgpt.com/
- https://null-byte.wonderhowto.com/collection/password-cracking/
- https://www.w3schools.com/cybersecurity/index.php