Password Complexity Checker

Evaluating Password Security Using Complexity Criteria and Real-Time Feedback

18 October 2024

Jagadish tripathy
Jagadishtripathy144@gmail.com

Table of Contents

Password Complexity Checker

1. Introduction

Passwords are one of the most commonly used mechanisms for securing access to systems, services, and data. However, weak passwords are often the cause of many security breaches. This report presents the design and implementation of a tool that assesses password strength based on established complexity criteria.

2. Objective

The objective of this tool is to evaluate the complexity of passwords entered by users based on criteria such as length, use of uppercase and lowercase letters, numbers, and special characters. The tool provides immediate feedback to users on the strength of their password and areas for improvement.

3. Password Complexity Criteria

To determine the complexity of a password, the tool uses the following criteria:

1. **Length**: The password must be at least 8 characters long.
2. **Uppercase Letters**: The password must contain at least one uppercase letter (A-Z).
3. **Lowercase Letters**: The password must contain at least one lowercase letter (a-z).
4. **Numbers**: The password must contain at least one digit (0-9).
5. **Special Characters**: The password must include at least one special character (e.g., !@#$%^&*()).

4. Design and Implementation

The tool was implemented using Python, utilizing regular expressions (re module) to check for each password criterion. Below is a summary of the design steps:

4.1. Design Steps

- **Step 1**: Check if the password has a length of at least 8 characters.
- **Step 2**: Verify if the password contains at least one lowercase letter.
- **Step 3**: Verify if the password contains at least one uppercase letter.
- **Step 4**: Check if there is at least one digit in the password.
- **Step 5**: Ensure the password contains at least one special character.
- **Step 6**: Based on how many criteria are met, assign a strength rating (Weak, Moderate, Strong).

4.2. Code Implementation

The Python code is structured to check each criterion and provide feedback to users. The implementation uses regular expressions to match patterns for letters, numbers, and special characters.

```
import re

def check_password_strength(password):
    length_criteria = len(password) >= 8
    lowercase_criteria = re.search(r'[a-z]', password) is not None
    uppercase_criteria = re.search(r'[A-Z]', password) is not None
    number_criteria = re.search(r'[0-9]', password) is not None
    special_char_criteria = re.search(r'[!@#$%^&*(),.?":{}|<>]', password) is not None

    criteria_met = sum([length_criteria, lowercase_criteria, uppercase_criteria, number_criteria,
special_char_criteria])

    if criteria_met == 5:
        strength = "Strong"
    elif 3 <= criteria_met < 5:
        strength = "Moderate"
    else:
        strength = "Weak"

    feedback = f"Password Strength: {strength}\n"
    feedback += f"- Length >= 8 characters: {'✓' if length_criteria else 'X'}\n"
    feedback += f"- Contains lowercase letter: {'✓' if lowercase_criteria else 'X'}\n"
    feedback += f"- Contains uppercase letter: {'✓' if uppercase_criteria else 'X'}\n"
    feedback += f"- Contains number: {'✓' if number_criteria else 'X'}\n"
    feedback += f"- Contains special character: {'✓' if special_char_criteria else 'X'}\n"

    return feedback
```

5. Testing and Results

The tool was tested with various passwords to verify that the criteria checks were functioning as expected. Below are the results for different test cases:

5.1. Test Case 1: Weak Password

- **Input**: abc123
- **Feedback**:
  - Password Strength: Weak
  - Length >= 8 characters: ✗
  - Contains lowercase letter: ✓
  - Contains uppercase letter: ✗
  - Contains number: ✓
  - Contains special character: ✗

5.2. Test Case 2: Moderate Password

- **Input**: Password123
- **Feedback:**
  - Password Strength**:** Moderate
  - Length >= 8 characters: ✓
  - Contains lowercase letter: ✓
  - Contains uppercase letter: ✓
  - Contains number: ✓
  - Contains special character: ✗

5.3. Test Case 3: Strong Password

- **Input**: StrongP@ssw0rd!
- **Feedback**:
  - Password Strength: Strong
  - Length >= 8 characters: ✓
  - Contains lowercase letter: ✓
  - Contains uppercase letter: ✓
  - Contains number: ✓
  - Contains special character: ✓

6. Conclusion

This Password Complexity Checker tool provides an efficient way for users to evaluate the strength of their passwords. By enforcing best practices, such as a combination of length, characters, and special symbols, the tool helps improve password security, reducing the likelihood of brute-force or dictionary attacks.

7. Recommendations for Improvement

To enhance the tool's functionality:

- **Password History Check**: Integrate a feature to prevent the reuse of previous passwords.
- **Entropy Calculation**: Implement password entropy calculation for a more robust strength assessment.
- **User-Friendly Interface**: Develop a graphical or web-based interface for non-technical users.