Password Strength Analyzer Project Report

By Kaustubha Nandakishore

Abstract

This report details the development and implementation of a Password Strength Analyzer, a minor cybersecurity project. The primary objective was to create a Python script that evaluates the security of a password based on a set of defined criteria. The script utilizes a scoring system to rate a password's strength as "Weak," "Medium," "Strong," or "Very Strong." It provides specific, actionable feedback to the user, guiding them to create more robust and secure passwords. The project serves as a practical application of fundamental programming concepts, including string manipulation, regular expressions, and logical conditions, within a cybersecurity context.

1. Introduction

In today's digital landscape, a password is the first line of defence against unauthorized access to personal and professional data. However, many users create passwords that are easy to guess or crack. The goal of this project was to develop a tool that not only assesses password strength but also educates the user on best practices. This Password Strength Analyzer aims to provide immediate, constructive feedback, thereby contributing to better password hygiene and enhanced online security.

2. Methodology

The project was developed using **Python**, leveraging its built-in libraries. The methodology focused on establishing a clear set of criteria for password strength and translating those into a logical scoring system.

2.1 Tools Used

- **Python:** The core programming language for the script.
- **re (Regular Expressions):** Used for efficient pattern matching to check for the presence of different character types.
- **getpass:** A secure module for handling password input without echoing the characters to the console.

2.2 Strength Criteria

A password's strength was determined by its compliance with the following five criteria. Each criterion met adds one point to the overall score.

1. **Length:** A minimum of 8 characters.
2. **Uppercase Letters:** Contains at least one uppercase letter (A-Z).
3. **Lowercase Letters:** Contains at least one lowercase letter (a-z).

4. **Numbers:** Contains at least one digit (0-9).
5. **Special Characters:** Contains at least one character that is not a letter or a number (e.g., !, @, #).

2.3 Scoring System

The final score determines the password's strength category:

- **Score 0-1: Weak**
- **Score 2: Medium**
- **Score 3-4: Strong**
- **Score 5: Very Strong**

3. Implementation

The script is structured around a main function (main) that handles user interaction and a core analysis function (check_password_strength).

The check_password_strength(password) function is the heart of the script. It initializes a score of zero and an empty list for feedback. It then sequentially checks the input password against each of the five criteria using if statements and regular expressions. For each criterion that is not met, a descriptive feedback string is added to the feedback list.

Regular Expressions used:

- r'[A-Z]': Checks for the presence of any uppercase letter.
- r'[a-z]': Checks for the presence of any lowercase letter.
- r'[0-9]': Checks for the presence of any digit.
- r'[^A-Za-z0-9]': Checks for the presence of any character that is not a letter or a digit, thus identifying a special character.

The main() function prompts the user for a password using getpass.getpass(), calls the analysis function, and prints a formatted output. It uses if/elif/else statements to categorize the password's strength based on the returned score and prints the relevant status and feedback. A while True loop allows the user to test multiple passwords in a single session.

4. Results and Analysis

The Password Strength Analyzer successfully fulfills its objective. The script correctly identifies the weaknesses in a given password and provides targeted feedback.

- **Example 1 (Weak):** pass123

    - Score: 2 (Length and Numbers)
    - Feedback: "Add at least one uppercase letter." "Add at least one special character."
- **Example 2 (Medium):** Password1
    - Score: 3 (Length, Uppercase, Lowercase, Numbers)
    - Feedback: "Add at least one special character."
- **Example 3 (Very Strong):** P@ssw0rd!
    - Score: 5 (All criteria met)
    - Feedback: "Your password is very strong! "

The feedback mechanism is highly effective, as it doesn't just state that a password is weak but explains precisely why and how to fix it. This educational aspect is crucial for promoting better security practices.

5. Conclusion

The Password Strength Analyzer project demonstrates a practical application of core programming principles to solve a common cybersecurity problem. The script is reliable, easy to use, and provides valuable feedback to the user. While this project is a solid foundation, there is room for future enhancements, such as:

- **Entropy Calculation:** Incorporating a more advanced entropy calculation to provide a more nuanced strength score.
- **Breached Password Check:** Checking the input password against a database of commonly used or breached passwords (e.g., the RockYou list).
- **GUI Interface:** Developing a basic graphical user interface using libraries like Tkinter to make the tool more user-friendly.

This project successfully met its objectives and provides a solid starting point for further exploration in cybersecurity.