BrooskieHub
Cybersecurity Internship Report

Task 10 : Password Strength Analyzer.

Submitted by: Namita Rana

Role: Cybersecurity Intern

Company: BrooskieHub

Date of Submission: 30 September 2025

Table of Contents

1. Introduction

1.1 Background of Password Security

Password security has become a central issue in modern cybersecurity. As digital platforms grow, the use of authentication systems based on passwords remains one of the most common security mechanisms. However, attackers are constantly developing sophisticated methods, including brute-force techniques and dictionary attacks, to exploit weak credentials. This makes it crucial to understand the role of strong passwords in safeguarding data.

The significance of password security extends beyond individual users. Large organizations rely on password-based authentication for both internal and external systems, meaning a single weak password could potentially compromise an entire network. Therefore, the study and implementation of tools that can assess password strength are critical in today's digital landscape.

1.2 Importance of Password Strength

Strong passwords serve as the first barrier against unauthorized access. A password that incorporates complexity—such as uppercase and lowercase letters, numbers, and special characters—makes it harder for attackers to guess or crack. By implementing strict password policies, organizations can reduce risks of data breaches and unauthorized access attempts.

Beyond technical strength, password complexity also influences user behaviour. Users educated on creating secure passwords are less likely to reuse them across platforms, reducing vulnerabilities. This project highlights the importance of building awareness around password strength and encourages safer practices for end-users.

1.3 Common Weaknesses in Passwords

Despite constant reminders, users often resort to predictable passwords such as "123456" or "password." Such choices significantly reduce the time it takes for automated tools to break into accounts. Attackers frequently leverage precompiled lists of these common passwords to exploit weak security measures.

Another major weakness arises from users reusing passwords across multiple accounts. Once an attacker compromises one service, they can attempt the same credentials on others, leading to widespread breaches. Simple structures, lack of special characters, and short length all contribute to the vulnerabilities of modern passwords.

2. Objective

2.1 Aim of the Task

The main aim of this task is to create a password strength analyser capable of evaluating passwords based on pre-established criteria. The tool checks for factors like minimum length, presence of uppercase and lowercase letters, digits, and special characters. By applying these rules, the analyser determines whether a password is weak, moderate, or strong.

Through this approach, the project not only evaluates passwords but also helps users understand why their chosen password is classified in a certain category. This aligns the project’s objective with real-world applications, where feedback mechanisms guide users to improve their password security.

2.2 Expected Outcomes

By completing this task, I expected to gain practical programming experience in Python while applying concepts of cybersecurity. The outcome includes a working tool that demonstrates how simple functions can address a serious security issue. This knowledge builds a foundation for future projects involving authentication and user data protection.

Another important outcome is user awareness. By interacting with the analyser, users are able to see in real-time how their password choices measure up to recommended standards. This educational aspect enhances the practical value of the project.

3. Tools and Environment

3.1 Programming Language: Python

Python was chosen as the programming language for its simplicity and wide usage in both academic and professional settings. With its clean syntax, Python allows developers to focus on problem-solving rather than complex coding structures. This makes it particularly suitable for beginners working on cybersecurity-related tasks.

In addition, Python provides a rich set of built-in string functions that are ideal for analysing password strength. Functions such as any(), isupper(), and isdigit() make it easy to verify different password requirements without relying on external libraries.

3.2 Development Platforms: VS Code and Jupyter Notebook

For development, I used **Visual Studio Code (VS Code)** due to its integrated features, debugging tools, and ability to handle Python projects efficiently. It also offers extensions that simplify coding and testing, making it a practical choice for building the analyser.

I also experimented with **Jupyter Notebook** because of its interactive environment. It allowed me to test code snippets individually, analyse results quickly, and document each step of the project, which proved helpful during development.

3.3 Supporting Libraries and Functions

Although external libraries were not strictly necessary, Python's built-in libraries provided most of the functionality I needed. For more advanced checks, I considered using the re module to integrate regular expressions, which would allow more detailed validations.

The flexibility of Python ensures that this project can be expanded in the future. For example, integrating external security libraries could help enforce stronger rules or even provide password entropy calculations.

4. Methodology

4.1 Rules for Password Strength

The analyser follows a structured set of rules to determine password strength. These include verifying that the password has a minimum length of eight characters, contains both uppercase and lowercase letters, includes at least one digit, and has one special character. These conditions align with widely accepted cybersecurity guidelines.

By enforcing these rules, the analyser ensures that passwords meet the baseline requirements for complexity. While the rules may not cover every advanced security measure, they create a foundation that significantly increases password resilience compared to weak alternatives.

4.2 Implementation Using Python Functions

Implementation relied heavily on Python's built-in string handling functions. For example, any(char.isupper() for char in password) was used to check the presence of uppercase letters, while any(char.isdigit() for char in password) verified digits. These concise checks made the code efficient and easy to read.

Combining these conditions into a logical sequence, the analyser outputs whether the password is weak, moderate, or strong. This modular design allowed me to adjust or extend the rules without rewriting the entire script, which makes the project adaptable for future improvements.

4.3 Input and Output Testing

Testing the analyser was an essential step to confirm that the rules worked as expected. I tested various weak passwords such as "mypassword" and "12345678," which the analyser correctly flagged as weak due to missing complexity. Similarly, intermediate passwords like "Pass1234" were classified as moderate.

Strong passwords such as "S@fep4ss123" passed all checks and were classified as strong, proving that the implementation worked correctly. These results validated the effectiveness of the analyser in guiding users toward creating secure credentials.

5. My Project

5.1 Project Design and Structure

The project was designed to be modular, with each rule implemented in a separate function. This not only enhanced readability but also made the code easier to maintain and expand. By keeping functions focused on specific tasks, debugging and testing were simplified significantly.

The project structure also emphasizes user interaction. The program prompts users for input, evaluates it against the rules, and then provides feedback. This interactive design makes the tool practical and educational at the same time.

5.2 Key Features and Functionality

One of the key features of my project is its ability to provide real-time feedback. Rather than just labelling passwords as weak or strong, the analyser highlights the specific requirements that were not met, such as missing special characters or insufficient length.

Another important functionality is the categorization of passwords into three levels—weak, moderate, and strong. This classification allows users to clearly understand where their password stands and what improvements are necessary, making the tool more useful in practical scenarios.

6. Implementation and Testing of My Project

6.1 Implementation Process

I implemented the password analyser by starting with basic checks like length and gradually adding more conditions for uppercase letters, digits, and special characters. This incremental approach ensured that each feature worked properly before moving on to the next.

During development, I also added user-friendly messages that explained why a password was rated as weak or moderate. This feature made the tool more than just a classifier—it became a learning aid for users.

6.2 Testing and Results

Testing involved running the program with different categories of passwords. Weak examples such as “password” and “abcdefg” were flagged correctly. Moderate examples like “Password1” passed some checks but failed others, showing accurate classification. Strong passwords like “Str0ng@Pass” were validated successfully.

The results confirmed that the tool functioned as intended. The testing phase also demonstrated the flexibility of the project, as it could be easily adapted to include additional rules if required in the future.

7. Outcome

7.1 Understanding Password Vulnerabilities

Through this project, I gained a deeper understanding of how weak passwords expose users to risks. The testing phase highlighted how quickly simple passwords can be classified as unsafe, reinforcing the importance of adopting stronger authentication measures.

This knowledge is directly applicable in real-world scenarios. By recognizing common weaknesses, developers and users alike can take proactive measures to strengthen their digital security.

7.2 Testing for Strong Passwords

The project also demonstrated the effectiveness of structured rules in guiding password creation. Strong passwords that fulfilled all requirements were consistently validated, showing the tool's reliability.

In practical use, such a tool could be integrated into registration forms to enforce password policies. This outcome shows that even simple Python scripts can have a significant impact on improving digital security practices.