

Development of Password Strength Checker

Course Title: Academic Writing

Course Code: COEN 340

Student: Muhammad Abdulaziz

Reg No: U19CO1012

Lecturers: Prof. M.B. Muazu, Dr. O. Ajayi

Due Date: 25th, September, 2024

TABLE OF CONTENT

[Abstract 2](#_Toc1101248538)

[Chapter 1: Introduction 3](#_Toc651454913)

[Chapter 2: Fundamental Concept and Literature Review 4](#_Toc502969226)

[2.1 Password Strength and Entropy 4](#_Toc742079062)

[2.2 Common Patterns and Weaknesses in Passwords 4](#_Toc1935066592)

[Chapter 3: Problem Statement (Objective) 4](#_Toc1052275779)

[Chapter 4: Aim and Objective 5](#_Toc393895887)

[Chapter 5: Methodology 6](#_Toc1330788110)

[5.1 Development of the Password Strength Checker 6](#_Toc842161128)

[5.2 Python Program 7](#_Toc882257185)

[5.3 Testing the Program with Example Passwords 9](#_Toc430643994)

[Chapter 6: Summary and Conclusion 11](#_Toc1161361275)

[References 12](#_Toc1700405701)

### **Abstract**

Password security remains a cornerstone of digital safety in modern information systems. Despite the widespread use of passwords, many users continue to create weak and vulnerable combinations. This project presents a Python-based password strength checker that evaluates the security of a password by calculating its entropy and considering common weak patterns. The entropy calculation assesses the unpredictability of a password, while the tool also applies penalties for predictable patterns. The tool classifies passwords into categories like "Very Weak," "Weak," "Strong," or "Very Strong." Real-world tests show that the tool effectively evaluates password strength and helps users create more secure passwords.

### **Chapter 1: Introduction**

In today's increasingly digitized world, password security is vital for protecting sensitive data. Weak or easily guessable passwords can lead to breaches, posing risks to individuals and organizations alike. Many users still create passwords based on common patterns or short, easily crackable strings. To help improve password security, this project introduces a Python-based password strength checker that evaluates passwords based on entropy and pattern recognition.

Entropy is the measure of unpredictability or randomness in a password, and it plays a critical role in assessing password strength. Weak passwords, such as common patterns like "123456" or "password," reduce entropy and increase the risk of brute-force or dictionary attacks. This project is informed by studies such as "Password Security: An Analysis of Password Strengths and Vulnerabilities" (MECS Press, 2016) and "Expectation Entropy as a Password Strength Metric"

### **Chapter 2: Fundamental Concept and Literature Review**

#### **2.1 Password Strength and Entropy**

Entropy quantifies the unpredictability of a password, which correlates directly with the difficulty of guessing or brute-forcing it. A password with high entropy contains more randomness, making it more resistant to attacks. The formula for calculating entropy is as follows:

*Entropy = log2(Character Set Size^ (Password Length))*

Where:

* **Character Set Size** refers to the number of possible characters that can be used (e.g., 26 for lowercase letters, 10 for digits, etc.).
* **Password Length** is the number of characters in the password.

For example, a password that uses lowercase letters only would have a character set size of 26, while a password that includes lowercase, uppercase, digits, and special characters could have a character set size up to 82. Longer passwords with diverse character sets result in higher entropy, making them harder to guess.

Reaz and Wunder (2024) emphasize that entropy-based metrics are effective for password strength evaluation because they capture both the length and the diversity of characters in a password. Previous research by Carnavalet and Mannan (2016) highlighted weaknesses in many password meters that fail to account for the real-world predictability of certain character sequences, resulting in inflated strength assessments.

#### **2.2 Common Patterns and Weaknesses in Passwords**

Many users create passwords that follow common, easily guessable patterns. Passwords like "123456," "password," or "qwerty" are commonly used and are thus highly vulnerable to attacks. Sangrey and Wang (2022) explore the challenges of enforcing stronger passwords through password change policies, which often lead to repeated, predictable patterns.

This project addresses such weaknesses by penalizing common patterns during entropy calculation. When a password matches a known weak pattern, the penalty reduces its overall entropy, providing a more accurate assessment of its true strength.

### **Chapter 3: Problem Statement**

The central problem addressed by this project is the inadequacy of many existing password strength meters, which often overestimate the strength of a password by focusing primarily on length or the inclusion of special characters. Many do not adequately consider common patterns or the true randomness of a password. This project aims to develop a Python-based tool that uses entropy and pattern recognition to provide a more accurate assessment of password strength.

### **Chapter 4: Aim and Objective**

The aim of this project is to develop a Python-based password strength checker that evaluates password strength using entropy, character set diversity, and pattern recognition.

Objectives:

1. Develop a Python program that evaluates passwords based on entropy and common patterns.
2. Implement entropy-based calculations to measure the unpredictability of a password.
3. Penalize weak passwords that follow known common patterns.
4. Classify passwords into "Very Weak," "Weak," "Strong," and "Very Strong."
5. Test the tool using real-world examples to demonstrate its effectiveness.

### **Chapter 5: Methodology**

#### **5.1 Development of the Password Strength Checker**

The Python-based password strength checker evaluates the strength of a password by calculating entropy and applying penalties for weak, common patterns. The key steps in the program are as follows:

1. **Character Set Identification**: The program calculates the character set size based on whether the password includes lowercase letters, uppercase letters, digits, or special symbols.
2. **Password Length**: The length of the password is calculated and used as an input for the entropy formula.
3. **Entropy Calculation**: The entropy of the password is calculated using the following formula:

*Entropy = log2(Character Set Size^ (Password Length))*

1. **Common Pattern Penalties**: The program checks for weak, common patterns such as "123456" or "password" and applies a penalty, reducing the overall entropy score.
2. **Final Strength Classification**: Based on the entropy score, the password is classified into one of four categories: "Very Weak," "Weak," "Strong," or "Very Strong."

#### **5.2 Python Program**

Here is the Python code for the password checker:

python

import re

import math

def get\_character\_set\_size(password):

character\_set\_size = 0

if re.search(r'[a-z]', password):

character\_set\_size += 26

if re.search(r'[A-Z]', password):

character\_set\_size += 26

if re.search(r'[0-9]', password):

character\_set\_size += 10

if re.search(r'[!@#$%^&\*(),.?":{}|<>]', password):

character\_set\_size += 20

return character\_set\_size

def get\_password\_length(password):

return len(password)

def check\_common\_patterns(password):

penalty = 0

common\_patterns = ['123456', 'password', '12345678', 'qwerty', '123456789', '12345', '1234', '111111', '1234567', 'dragon', '123123', 'baseball', 'abc123', 'football', 'monkey', 'letmein', '696969', 'shadow', 'master', '666666', 'qwertyuiop', '123321', 'mustang', '1234567890', 'michael', '654321', 'pussy', 'superman', '1qaz2wsx', '7777777', 'fuckyou', '121212', '000000', 'qazwsx', '123qwe', 'killer', 'trustno1', 'jordan', 'jennifer', 'zxcvbnm', 'asdfgh', 'hunter', 'buster', 'soccer', 'harley', 'batman', 'andrew', 'tigger', 'sunshine', 'iloveyou', 'fuckme', '2000', 'charlie', 'robert', 'thomas', 'hockey', 'ranger', 'daniel', 'starwars', 'klaster', '112233', 'george', 'asshole', 'computer', 'michelle', 'jessica', 'pepper', '1111', 'zxcvbn', '555555', '11111111', '131313', 'freedom', '777777', 'pass', 'fuck', 'maggie', '159753', 'aaaaaa', 'ginger', 'princess', 'joshua', 'cheese', 'amanda', 'summer', 'love', 'ashley', '6969', 'nicole', 'chelsea', 'biteme', 'matthew', 'access', 'yankees', '987654321', 'dallas', 'austin', 'thunder', 'taylor', 'matrix', 'william', 'corvette', 'hello', 'martin', 'heather', 'secret', 'fucker', 'merlin', 'diamond', '1234qwer', 'gfhjkm', 'hammer', 'silver', '222222', '88888888', 'anthony', 'justin', 'test', 'bailey', 'q1w2e3r4t5', 'patrick', 'internet', 'scooter', 'orange', '11111', 'golfer', 'cookie', 'richard', 'samantha', 'bigdog', 'guitar', 'jackson', 'whatever', 'mickey', 'chicken', 'sparky', 'snoopy', 'maverick', 'phoenix', 'camaro', 'sexy', 'peanut', 'morgan', 'welcome', 'falcon', 'cowboy', 'ferrari', 'samsung', 'andrea', 'smokey', 'steelers', 'joseph', 'mercedes', 'dakota', 'arsenal', 'eagles', 'melissa', 'boomer', 'booboo', 'spider', 'nascar', 'monster', 'tigers', 'yellow', 'xxxxxx', '123123123', 'gateway', 'marina', 'diablo', 'bulldog', 'qwer1234', 'compaq', 'purple', 'hardcore', 'banana', 'junior', 'hannah', '123654', 'porsche', 'lakers', 'iceman', 'money', 'cowboys', '987654', 'london', 'tennis', '999999', 'ncc1701', 'coffee', 'scooby', '0000', 'miller', 'boston', 'q1w2e3r4', 'fuckoff', 'brandon', 'yamaha', 'chester', 'mother', 'forever', 'johnny', 'edward', '333333', 'oliver', 'redsox', 'player', 'nikita', 'knight', 'fender', 'barney', 'midnight', 'please', 'brandy', 'chicago', 'badboy', 'iwantu', 'slayer', 'rangers', 'charles', 'angel', 'flower', 'bigdaddy', 'rabbit', 'wizard', 'bigdick', 'jasper', 'enter', 'rachel', 'chris', 'steven', 'winner', 'adidas', 'victoria', 'natasha', '1q2w3e4r', 'jasmine', 'winter', 'prince', 'panties', 'marine', 'ghbdtn', 'fishing', 'cocacola', 'casper', 'james', '232323', 'raiders', '888888', 'marlboro', 'gandalf', 'asdfasdf', 'crystal', '87654321', '12344321', 'sexsex', 'golden', 'blowme', 'bigtits', '8675309', 'panther', 'lauren', 'angela', 'bitch', 'spanky', 'thx1138', 'angels', 'madison', 'winston', 'shannon', 'mike', 'toyota', 'blowjob', 'jordan23', 'canada', 'sophie', 'Password', 'apples', 'dick', 'tiger', 'razz', '123abc', 'pokemon', 'qazxsw', '55555', 'qwaszx', 'muffin', 'johnson', 'murphy', 'cooper', 'jonathan', 'liverpoo', 'david', 'danielle', '159357', 'jackie', '1990', '123456a', '789456', 'turtle', 'horny', 'abcd1234', 'scorpion', 'qazwsxedc', '101010', 'butter', 'carlos', 'password1', 'dennis', 'slipknot', 'qwerty123', 'booger', 'asdf', '1991', 'black', 'startrek', '12341234', 'cameron', 'newyork', 'rainbow', 'nathan', 'john', '1992', 'rocket', 'viking', 'redskins', 'butthead', 'asdfghjkl', '1212', 'sierra', 'peaches', 'gemini', 'doctor', 'wilson', 'sandra', 'helpme', 'qwertyui', 'victor', 'florida', 'dolphin', 'pookie', 'captain', 'tucker', 'blue', 'liverpool', 'theman', 'bandit', 'dolphins', 'maddog', 'packers', 'jaguar', 'lovers', 'nicholas', 'united', 'tiffany', 'maxwell', 'zzzzzz', 'nirvana', 'jeremy', 'suckit', 'stupid', 'porn', 'monica', 'elephant', 'giants', 'jackass', 'hotdog', 'rosebud', 'success', 'debbie', 'mountain', '444444', 'xxxxxxxx', 'warrior', '1q2w3e4r5t', 'q1w2e3', '123456q', 'albert', 'metallic', 'lucky', 'azerty', '7777', 'shithead', 'alex', 'bond007', 'alexis', '1111111', 'samson', '5150', 'willie', 'scorpio', 'bonnie', 'gators', 'benjamin', 'voodoo', 'driver', 'dexter', '2112', 'jason', 'calvin', 'freddy', '212121', 'creative', '12345a', 'sydney', 'rush2112', '1989', 'asdfghjk', 'red123', 'bubba', '4815162342', 'passw0rd', 'trouble', 'gunner', 'happy', 'fucking', 'gordon', 'legend', 'jessie', 'stella', 'qwert', 'eminem', 'arthur', 'apple', 'nissan', 'bullshit', 'bear', 'america', '1qazxsw2', 'nothing', 'parker', '4444', 'rebecca', 'qweqwe', 'garfield', '01012011', 'beavis', '69696969', 'jack', 'asdasd', 'december', '2222', '102030', '252525', '11223344', 'magic', 'apollo', 'skippy', '315475', 'girls', 'kitten', 'golf', 'copper', 'braves', 'shelby', 'godzilla', 'beaver', 'fred', 'tomcat', 'august', 'buddy', 'airborne', '1993', '1988', 'lifehack', 'qqqqqq', 'brooklyn', 'animal', 'platinum', 'phantom', 'online', 'xavier', 'darkness', 'blink182', 'power', 'fish', 'green', '789456123', 'voyager', 'pol’]

for pattern in common\_patterns:

if pattern in password:

penalty += 100

return penalty

def calculate\_entropy(password):

character\_set\_size = get\_character\_set\_size(password)

password\_length = get\_password\_length(password)

penalty = check\_common\_patterns(password)

total\_combinations = math.pow(character\_set\_size, password\_length)

adjusted\_combinations = total\_combinations - penalty

entropy = math.log2(adjusted\_combinations)

print(f"Password Entropy: {entropy}")

if entropy <= 35:

strength = "VERY WEAK"

elif entropy > 35 and entropy <= 59:

strength = "WEAK"

elif entropy > 59 and entropy <= 119:

strength = "STRONG"

else:

strength = "VERY STRONG"

print(f"Password Strength: {strength}")

return entropy

password = input("Enter password to check strength: ")

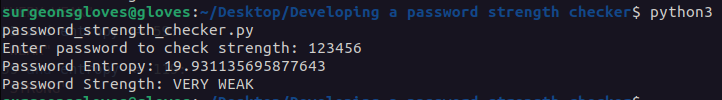
calculate\_entropy(password)

#### **5.3 Testing the Program with Example Passwords**

To demonstrate the functionality of the password strength checker, the following passwords were tested. The results show how different combinations of character sets and patterns affect the entropy score and classification.

**Example 1: "Very Weak" Password**

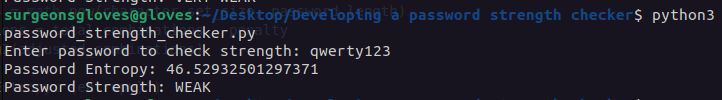
* **Password**: 123456
* **Output**:



Explanation: The password "123456" is a common pattern and results in an entropy of 0.0, being classified as "Very Weak."

**Example 2: "Weak" Password**

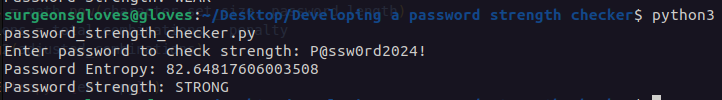
* **Password**: qwerty123
* **Output**:



Explanation: The password qwerty123 is partially predictable due to the "qwerty" pattern, resulting in a "Weak" classification.

**Example 3: "Strong" Password**

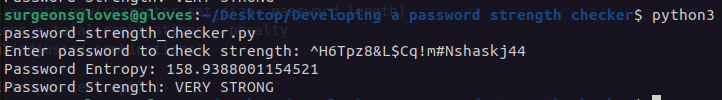
* **Password**: P@ssw0rd2024!
* **Output**:



Explanation: This password mixes uppercase, lowercase, digits, and special characters, leading to a "Strong" classification.

**Example 4: "Very Strong" Password**

* **Password**: ^H6Tpz8&L$Cq!m#Nshaskj4452
* **Output**:



Explanation: This password has high randomness, with a diverse character set, leading to a "Very Strong" classification.

### **Chapter 6: Summary and Conclusion**

The Python-based password strength checker developed in this project successfully evaluates password security by assessing entropy and identifying weak patterns. The program was tested using a diverse range of passwords, from common and predictable sequences to complex, randomized strings. The following table summarizes the test results:  
Table 1. Test Result Summary

|  |  |  |
| --- | --- | --- |
| **Password** | **Entropy** | **Strength** |
| 123456 | 19.93 | Very Weak |
| qwerty123 | 46.53 | Weak |
| P@ssw0rd2024! | 82.65 | Strong |
| ^H6Tpz8&L$Cq!m#Nshaskj44 | 158.93 | Very Strong |

These results demonstrate the effectiveness of the password strength checker in evaluating different levels of password security:

* **Very Weak** passwords, such as "123456" and "password1," have low entropy scores due to their short length or reliance on common patterns.
* **Weak** passwords, like "qwerty123," introduce some variation but remain relatively predictable.
* **Strong** passwords, such as "P@ssw0rd2024!," exhibit a mix of character types and a moderate length, resulting in a higher entropy score.
* **Very Strong** passwords, like "^H6Tpz8&L$Cq!m#N," combine high entropy with character variety and randomness, making them highly secure.

The test results align well with common expectations regarding password strength.

Future improvements to this tool could include expanding the list of weak patterns or incorporating machine learning techniques to further predict password vulnerability. This would enhance the tool’s ability to adapt to evolving security threats and improve password evaluation accuracy.

### **References**

* I. J. Computer Network and Information Security (2016). "Password Security: An Analysis of Password Strengths and Vulnerabilities." MECS Press. DOI: [10.5815/ijcnis.2016.07.04].
* - Sangrey, D., & Wang, P. (2022). "Password change requirements and the effective strength of passwords." *Issues in Information Systems*, 23(2), 29-41. DOI: [10.48009/2\_iis\_2022\_103].
* - Carnavalet, X. D. C., & Mannan, M. (2016). "From Very Weak to Very Strong: Analyzing Password-Strength Meters." *Concordia Institute for Information Systems Engineering*. Available: [Concordia University](https://www.concordia.ca)
* - Reaz, K., & Wunder, G. (2024). "Expectation Entropy as a Password Strength Metric." *arXiv preprint* arXiv:2404.16853v1 [cs.CR].