# Password Strength Checker and Brute-Force Estimation Tool

- Submitted by: Srushti Mhatre & Aisha Shaikh
- Cybersecurity Internship – 2025
- Digisuraksha Parhari Foundation
- Infinisec Technologies Pvt. Ltd.
- Date: 12th May 2025

## Introduction

- This project, developed during the Cybersecurity Internship at the Digisuraksha Parhari Foundation, focuses on creating a simple tool that educates users on password strength and the time it would take to crack passwords using brute-force techniques.

## Abstract

- Passwords are the most common form of user authentication but are often the weakest link due to poor choices by users. This project introduces a lightweight Python tool that checks for password strength using basic heuristics such as length, character variety, and presence in common password lists. It also estimates the time required to crack passwords using brute-force at a rate of 1 million guesses per second, providing practical insight into how vulnerable a weak password can be.

## Problem Statement

- A significant portion of cyberattacks result from weak or reused passwords. Many users opt for simple and predictable passwords. This tool addresses this issue by analyzing a password’s structure and educating users about the risks associated with weak credentials. The ultimate goal is to promote strong password hygiene and cybersecurity awareness.

## Literature Review

- Several studies and industry reports have highlighted the dangers of weak passwords. The Verizon Data Breach Report and Microsoft’s findings emphasize that stolen or reused credentials are involved in most breaches. Guidelines by organizations like NIST and OWASP recommend using strong, unique passwords. Tools like John the Ripper and data from password dumps (like RockYou.txt) have exposed the vulnerabilities of weak passwords, reinforcing the need for such educational tools.

## Research Methodology

- The project used Python to create a command-line tool that analyzes passwords. It checks for different criteria, including password length and the use of various character types (uppercase, lowercase, digits, and symbols). For brute-force estimation, it calculates the total number of possible combinations based on the character set used, then divides by an assumed cracking rate to estimate how long a password would take to be guessed.

# Tool Implementation

- The implementation consists of two main Python functions:
- 1. check_strength() evaluates the password based on character types and flags weaknesses.
- 2. estimate_brute_force_time() calculates the estimated time to brute-force the password using simple mathematical logic.
- The tool runs through the command line and doesn’t require any external libraries, making it lightweight and easy to use.

## Results & Observations

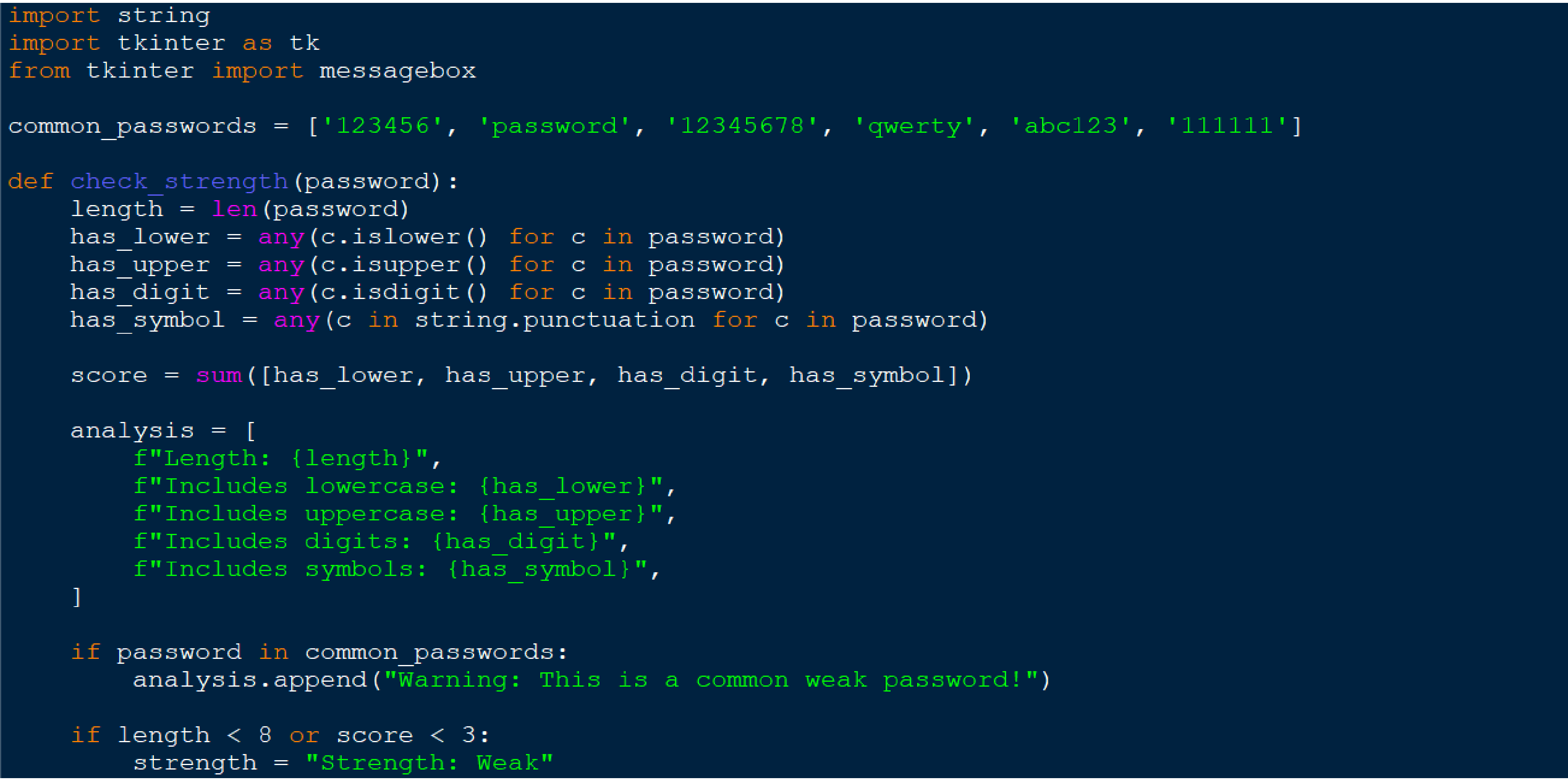
- Tests showed that passwords shorter than 8 characters and those using predictable patterns were easy to crack. Conversely, passwords that included a mix of letters, numbers, and symbols—and were longer—significantly increased the time required for brute-force cracking. This supports best practices recommended by security experts.

## Ethical Use & Relevance

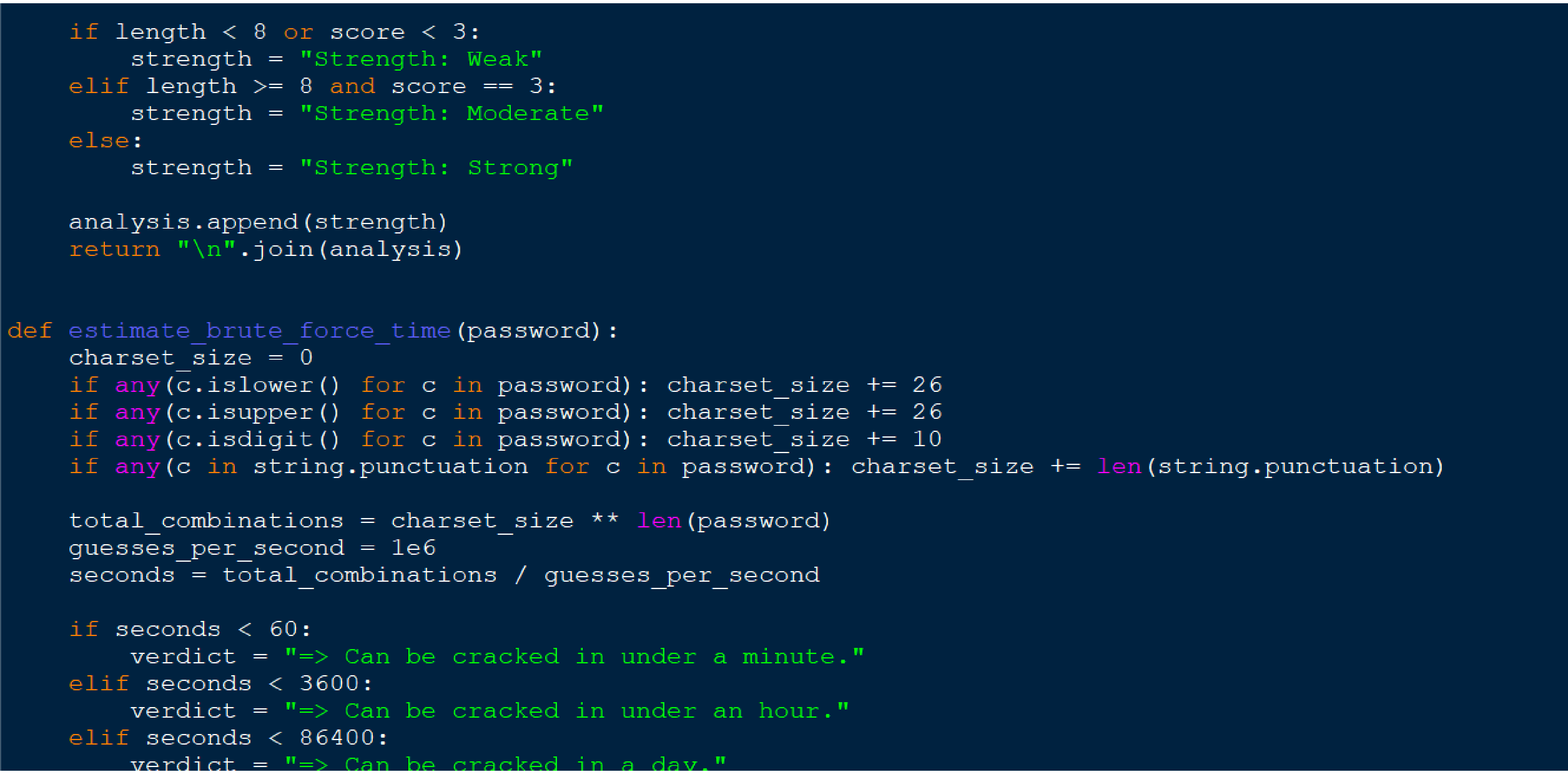
- This tool is designed for ethical use only. It’s ideal for personal training, workshops, and classroom education. By visualizing how quickly a password could be cracked, it encourages users to adopt stronger, more secure practices in their daily digital lives without instilling fear or encouraging misuse.

## Future Scope

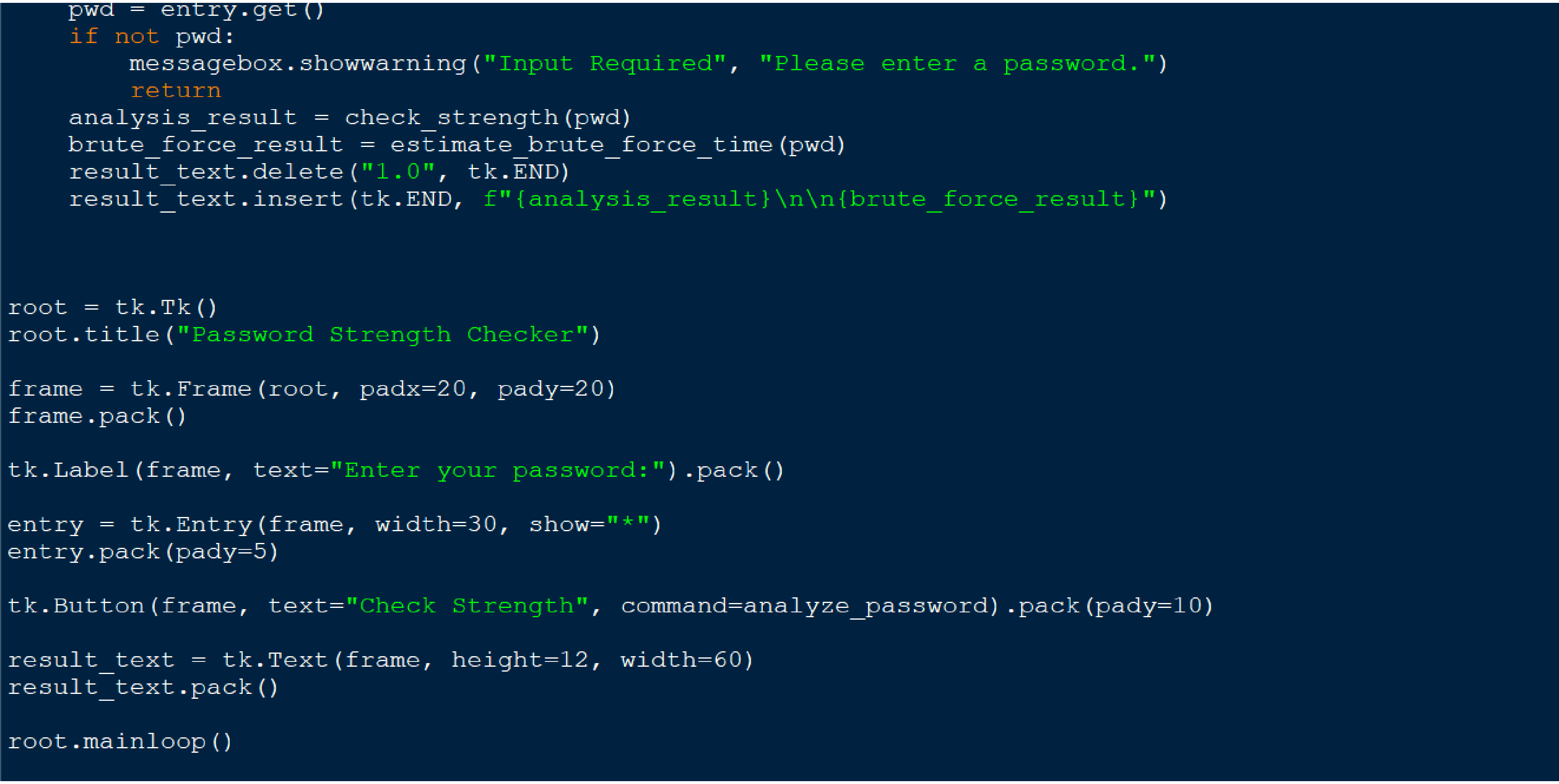
- Potential enhancements to the tool include:
- - Adding a Graphical User Interface (GUI) for easier use.
- - Integration with password managers to offer real-time feedback.
- - Logic to detect keyboard patterns (like “qwerty” or “12345”).
- - Simulation of dictionary attacks to provide a more accurate risk assessment.



```python
import string
import tkinter as tk
from tkinter import messagebox

common_passwords = ['123456', 'password', '12345678', 'qwerty', 'abc123', '111111']

def check_strength(password):
    length = len(password)
    has_lower = any(c.islower() for c in password)
    has_upper = any(c.isupper() for c in password)
    has_digit = any(c.isdigit() for c in password)
    has_symbol = any(c in string.punctuation for c in password)

    score = sum([has_lower, has_upper, has_digit, has_symbol])

    analysis = [
        f"Length: {length}",
        f"Includes lowercase: {has_lower}",
        f"Includes uppercase: {has_upper}",
        f"Includes digits: {has_digit}",
        f"Includes symbols: {has_symbol}",
    ]

    if password in common_passwords:
        analysis.append("Warning: This is a common weak password!")

    if length < 8 or score < 3:
        strength = "Strength: Weak"
```



```python
    if length < 8 or score < 3:
        strength = "Strength: Weak"
    elif length >= 8 and score == 3:
        strength = "Strength: Moderate"
    else:
        strength = "Strength: Strong"

    analysis.append(strength)
    return "\n".join(analysis)


def estimate_brute_force_time(password):
    charset_size = 0
    if any(c.islower() for c in password): charset_size += 26
    if any(c.isupper() for c in password): charset_size += 26
    if any(c.isdigit() for c in password): charset_size += 10
    if any(c in string.punctuation for c in password): charset_size += len(string.punctuation)

    total_combinations = charset_size ** len(password)
    guesses_per_second = 1e6
    seconds = total_combinations / guesses_per_second

    if seconds < 60:
        verdict = "=> Can be cracked in under a minute."
    elif seconds < 3600:
        verdict = "=> Can be cracked in under an hour."
    elif seconds < 86400:
        verdict = "=> Can be cracked in a day."
```



```
    pwd = entry.get()
    if not pwd:
        messagebox.showwarning("Input Required", "Please enter a password.")
        return
    analysis_result = check_strength(pwd)
    brute_force_result = estimate_brute_force_time(pwd)
    result_text.delete("1.0", tk.END)
    result_text.insert(tk.END, f"{analysis_result}\n\n{brute_force_result}")



root = tk.Tk()
root.title("Password Strength Checker")

frame = tk.Frame(root, padx=20, pady=20)
frame.pack()

tk.Label(frame, text="Enter your password:").pack()

entry = tk.Entry(frame, width=30, show="*")
entry.pack(pady=5)

tk.Button(frame, text="Check Strength", command=analyze_password).pack(pady=10)

result_text = tk.Text(frame, height=12, width=60)
result_text.pack()

root.mainloop()
```

## GITHUB LINK:

https://github.com/srushtimhatre29/password_strength_checker_tool.git

## DEMO LINK:

https://github.com/srushtimhatre29/password_strength_checker_tool/blob/main/demo/demo_video.mp4

# Conclusion

- The Password Strength Checker and Brute-Force Estimation Tool provides a valuable resource for educating users on the importance of password security. It reinforces best practices and serves as a foundational step toward broader cybersecurity awareness.