Task-3 Password Complexity Checker

Report By

Leburi SriRam

Intern At

Prodigy Infotech

July 11,2024

Table of Contents:

Abstract

Introduction

Understanding password standards

Program

Conclusion

Reference

Abstract:

To build a tool to check the strength of a password based on criteria such as length,presence of upper case letters,lower case letters and special characters. The primary objective is to enhance user security by encouraging the creation of robust passwords through real-time feedback. By integrating a user-friendly interface with comprehensive strength assessment algorithms, this tool aims to educate users on the essential components of a strong password, thereby reducing the risk of unauthorized access and improving overall cybersecurity hygiene.

Introduction:

In today's digital age, the security of online accounts and personal data is critically dependent on the strength of passwords. Weak or easily guessable passwords are a common vulnerability that can lead to unauthorized access and data breaches. Despite widespread awareness of the importance of strong passwords, many users continue to rely on simple, insecure passwords due to a lack of understanding or convenience.

This project introduces a Password Strength Assessment Tool, a graphical user interface (GUI) application that aids users in creating secure passwords. Utilizing Python and the tkinter library, this tool evaluates passwords against a set of criteria known to contribute to password strength: length, the inclusion of uppercase and lowercase letters, numbers, and special characters. By providing immediate feedback on the strength of the entered password and specific suggestions for improvement, the tool aims to educate users on creating stronger passwords.

The development of this tool addresses the need for accessible and straightforward mechanisms for password evaluation. It serves not only as a practical utility for individual users but also as a potential educational resource for organizations aiming to enhance their employees' cybersecurity practices. This introduction outlines the significance of password strength in the context of cybersecurity and sets the stage for a detailed description of the tool's functionality and implementation.

Requirements:

System or PC

Jupyter notebook or any interpreter

Tkinter package should be installed

Understanding Password Security standards

Password security standards:

Password security standards are guidelines and best practices designed to help users create strong passwords that are resistant to various forms of attacks, such as brute-force attacks, dictionary attacks, and social engineering. Below are some widely recognized password security standards and practices:

Common Password Security Standards

Length:

Passwords should be at least 12 characters long. Longer passwords are generally more secure because they increase the number of possible combinations.

Complexity:

Passwords should include a mix of uppercase letters, lowercase letters, numbers, and special characters (e.g., !@#$%^&*(),.?":{}|<>).

Avoid Common Words and Patterns:

Passwords should not contain easily guessable information such as common words, phrases, or patterns (e.g., "password123", "qwerty", "abc123").

Unique Passwords:

Each account should have a unique password to prevent a breach in one system from compromising others.

No Personal Information:

Passwords should not include easily obtainable personal information such as names, birthdays, or addresses.

Advanced Password Security Practices

Multi-Factor Authentication (MFA):

Use MFA to add an extra layer of security. This requires users to provide two or more verification factors to gain access to a resource.

Password Managers:

Use password managers to generate, store, and manage complex passwords securely. This helps users maintain unique passwords for different accounts without needing to remember each one.

Regular Changes:

Change passwords regularly, especially if there is a suspicion that they may have been compromised.

Password Expiration:

Implement password expiration policies where users are required to update their passwords periodically (e.g., every 90 days).

Account Lockout Mechanism:

Implement account lockout mechanisms that temporarily disable accounts after a certain number of failed login attempts to prevent brute-force attacks.

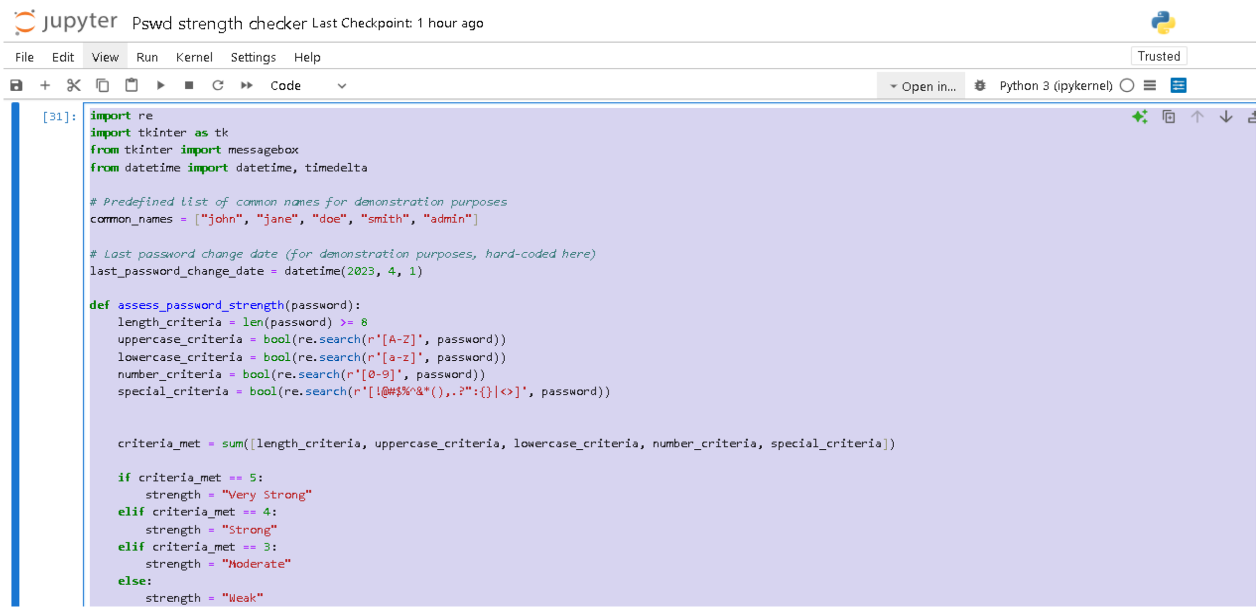
Encryption:

Store passwords securely using strong encryption algorithms. Avoid storing passwords in plain text

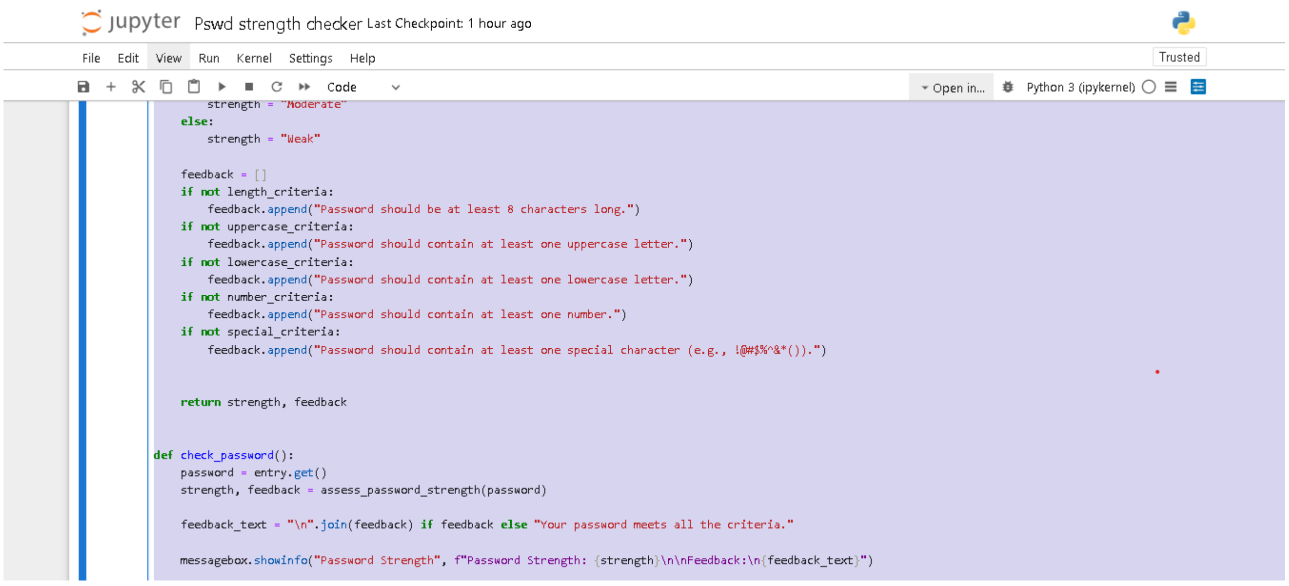
Python Program:

```
import re

import tkinter as tk

from tkinter import messagebox

from datetime import datetime, timedelta


# Predefined list of common names for demonstration purposes

common_names = ["john", "jane", "doe", "smith", "admin"]


# Last password change date (for demonstration purposes, hard-coded here)

last_password_change_date = datetime(2023, 4, 1)


def assess_password_strength(password):

    length_criteria = len(password) >= 8

    uppercase_criteria = bool(re.search(r'[A-Z]', password))

    lowercase_criteria = bool(re.search(r'[a-z]', password))

    number_criteria = bool(re.search(r'[0-9]', password))

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



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


    if criteria_met == 5:

        strength = "Very Strong"

    elif criteria_met == 4:
        strength = "Strong"
    elif criteria_met == 3:
        strength = "Moderate"
    else:
        strength = "Weak"

    feedback = []
    if not length_criteria:
        feedback.append("Password should be at least 8 characters long.")
    if not uppercase_criteria:
        feedback.append("Password should contain at least one uppercase letter.")
    if not lowercase_criteria:
        feedback.append("Password should contain at least one lowercase letter.")
    if not number_criteria:
        feedback.append("Password should contain at least one number.")
    if not special_criteria:
        feedback.append("Password should contain at least one special character (e.g.,
!@#$%^&*()).")



    return strength, feedback



def check_password():
    password = entry.get()
    strength, feedback = assess_password_strength(password)
    feedback_text = "\n".join(feedback) if feedback else "Your password meets all the criteria."

    messagebox.showinfo("Password Strength", f"Password Strength: {strength}\n\nFeedback:\n{feedback_text}")

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

frame = tk.Frame(root, padx=20, pady=20)
frame.pack(padx=10, pady=10)

label = tk.Label(frame, text="Enter your password:")
label.pack(pady=5)

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

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

root.mainloop()
```

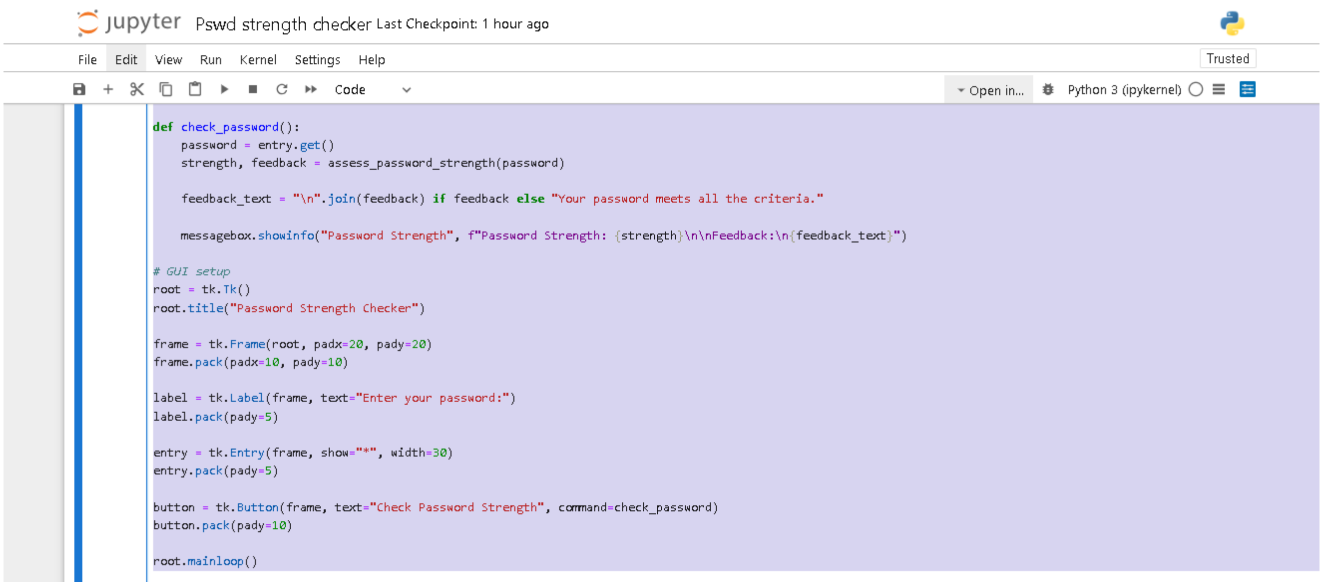
Screen shots:


jupyter Pswd strength checker Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)

```
[31]: import re
      import tkinter as tk
      from tkinter import messagebox
      from datetime import datetime, timedelta

      # Predefined list of common names for demonstration purposes
      common_names = ["john", "jane", "doe", "smith", "admin"]

      # Last password change date (for demonstration purposes, hard-coded here)
      last_password_change_date = datetime(2023, 4, 1)

      def assess_password_strength(password):
          length_criteria = len(password) >= 8
          uppercase_criteria = bool(re.search(r'[A-Z]', password))
          lowercase_criteria = bool(re.search(r'[a-z]', password))
          number_criteria = bool(re.search(r'[0-9]', password))
          special_criteria = bool(re.search(r'[!@#$%^&*(),.?":{}|<>]', password))


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

          if criteria_met == 5:
              strength = "Very Strong"
          elif criteria_met == 4:
              strength = "Strong"
          elif criteria_met == 3:
              strength = "Moderate"
          else:
              strength = "Weak"
```

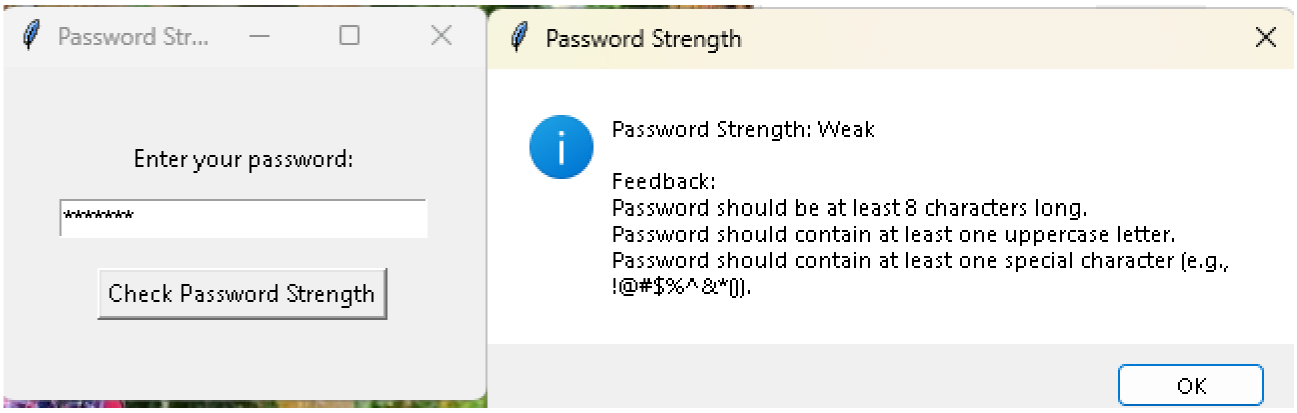

jupyter Pswd strength checker Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)

```
            strength = "Moderate"
        else:
            strength = "Weak"

        feedback = []
        if not length_criteria:
            feedback.append("Password should be at least 8 characters long.")
        if not uppercase_criteria:
            feedback.append("Password should contain at least one uppercase letter.")
        if not lowercase_criteria:
            feedback.append("Password should contain at least one lowercase letter.")
        if not number_criteria:
            feedback.append("Password should contain at least one number.")
        if not special_criteria:
            feedback.append("Password should contain at least one special character (e.g., !@#$%^&*()).")


        return strength, feedback


    def check_password():
        password = entry.get()
        strength, feedback = assess_password_strength(password)

        feedback_text = "\n".join(feedback) if feedback else "Your password meets all the criteria."

        messagebox.showinfo("Password Strength", f"Password Strength: {strength}\n\nFeedback:\n{feedback_text}")
```

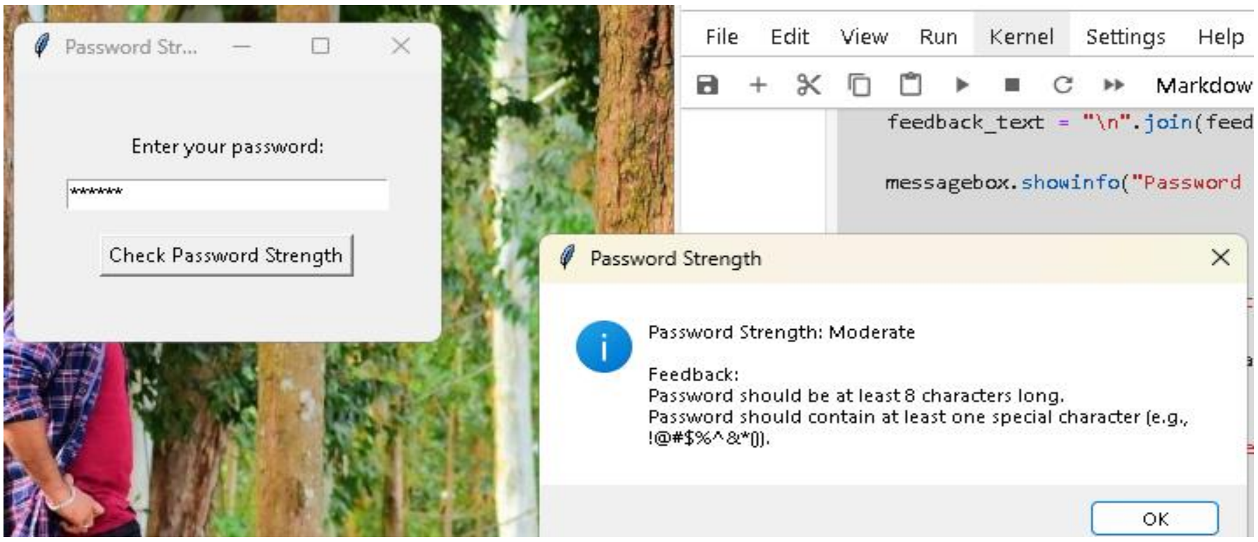

jupyter Pswd strength checker Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in...
Python 3 (ipykernel)

```
def check_password():
    password = entry.get()
    strength, feedback = assess_password_strength(password)

    feedback_text = "\n".join(feedback) if feedback else "Your password meets all the criteria."

    messagebox.showinfo("Password Strength", f"Password Strength: {strength}\n\nFeedback:\n{feedback_text}")

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

frame = tk.Frame(root, padx=20, pady=20)
frame.pack(padx=10, pady=10)

label = tk.Label(frame, text="Enter your password:")
label.pack(pady=5)

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

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

root.mainloop()
```

Output:

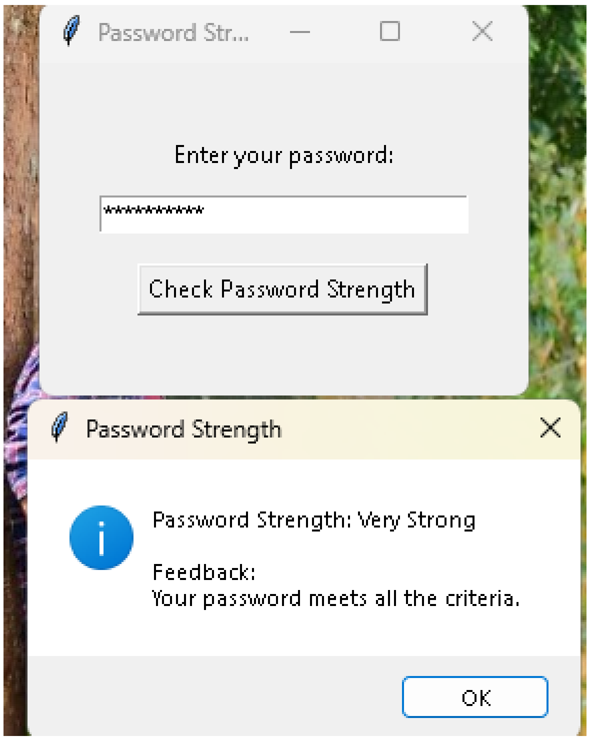


```
Password Strength: Weak

Feedback:
Password should be at least 8 characters long.
Password should contain at least one uppercase letter.
Password should contain at least one special character (e.g.,
!@#$%^&*()).
```



```
feedback_text = "\n".join(feed

messagebox.showinfo("Password
```

```
Password Strength: Moderate

Feedback:
Password should be at least 8 characters long.
Password should contain at least one special character (e.g.,
!@#$%^&*()).
```



```
Password Strength: Very Strong

Feedback:
Your password meets all the criteria.
```

Conclusion:

This comprehensive guide has outlined the developed Password Strength Checker application using Python and Tkinter GUI toolkit. The goal was to assess the strength of user-entered passwords based on various criteria and provide feedback to help users create strong and secure passwords.

In conclusion, the Password Strength Checker application provides a practical solution for users aiming to create strong and secure passwords. By offering real-time feedback and suggesting improvements, the application promotes better password practices and contributes to overall digital security awareness.

During this task allotted by Prodigy Infotech. I have gained valuable insights into importance of password, password standards and policies in securing data.And some new knowledge I gained in depth about the security standards and practices involved in this.The support and resources provided by Prodigy infotech have been instrumental in enhancing my understanding and skills in Cyber security.

Reference:

https://itsecurityhq.com/understanding-password-policy-why-its-important-and-best-practices-to-follow/

https://learn.microsoft.com/en-us/microsoft-365/admin/misc/password-policy-recommendations?view=o365-worldwide