Password Strength Analyzer with Custom Wordlist Generator

Name: Kurma Sampath
Submission Date: 27-06-2025

Abstract

This project focuses on developing a **command-line tool** using Python that analyzes the strength of user-provided passwords and generates custom wordlists based on personal information like name, pet name, and date of birth. These wordlists are commonly used in cybersecurity fields for password cracking simulations, auditing, and penetration testing.

Objectives

- To analyze password strength based on complexity criteria
- To collect user inputs (name, pet name, birth year) and generate a personalized wordlist
- To include patterns such as leetspeak, appending numbers, capitalization, and reversals
- To save the wordlist in .txt format for use in password testing tools
- To implement the tool using command-line arguments in VS Code

Tools & Technologies Used

Tool/Technology	**Purpose**
Python 3.11	Programming Language
Visual Studio Code	Code Editor and Execution
argparse Module	Parsing command-line arguments
string Module	Character classification
Terminal / CMD	Tool execution

1. Password Strength Analyzer

- Evaluates password length and complexity
- Checks for uppercase, lowercase, digits, and symbols

- Provides score (0–4) and suggestions to improve password strength

```
python main.py --analyze Pass@123 --name elevateLabs --pet elevate --dob 2005
```

```
--- Password Strength Analysis ---
Length: 8
Complexity Score (0-4): 4
Strong password ✅

--- Wordlist Generator ---
✅ Wordlist saved as 'wordlist.txt'
PS C:\Users\DELL\OneDrive\Desktop\proj>
```

2. Custom Wordlist Generator

- Takes: --name, --pet, --dob from command-line
- Generates word variations like:
  - name123, pet2024, name@pet, reverse strings, leetspeak
- Stores the output in wordlist.txt for use in password testing tools

```
3l3v@t3l@b$
elevatelabs123
2005@elevatelabs
elevate!
etavele
2005!
elevatelabs
elevatelabs@elevatelabs
elevate2024
elevate@elevatelabs
elevate
3l3v@t3
elevatelabs!
elevatelabs2024
elevate123
2005123
5002
2005
Elevatelabs
20052024
sbaletavele
Elevate
```

3. Command-Line Interface (Using argparse)

- Tool is executed via terminal:

python main.py --analyze Pass@123 --name elevateLabs --pet elevate --dob 2005

Returns strength analysis and saves wordlist

4. main.py

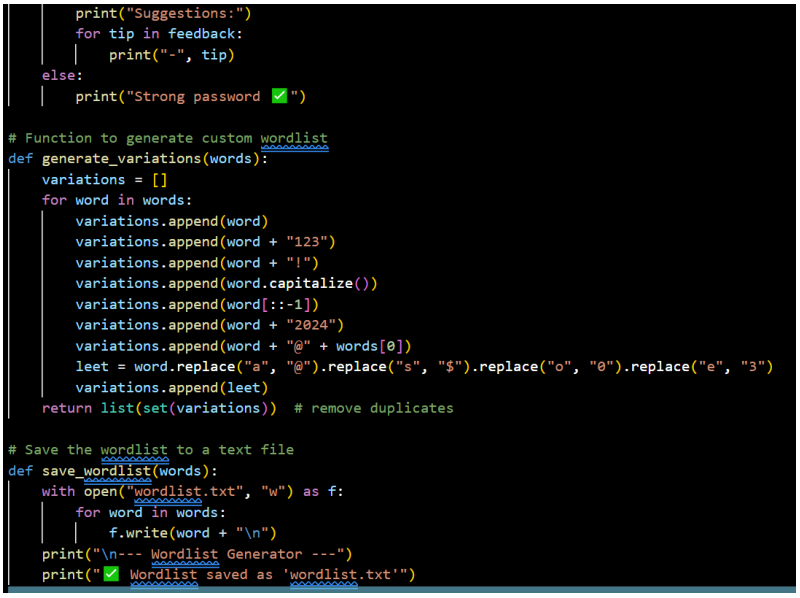
```python
import argparse
import string

# Function to analyze the password
def analyze_password(password):
    length = len(password)
    has_upper = any(c.isupper() for c in password)
    has_lower = any(c.islower() 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_upper, has_lower, has_digit, has_symbol])
    feedback = []

    if length < 8:
        feedback.append("Password is too short.")
    if not has_upper:
        feedback.append("Add uppercase letters.")
    if not has_lower:
        feedback.append("Add lowercase letters.")
    if not has_digit:
        feedback.append("Add numbers.")
    if not has_symbol:
        feedback.append("Add special characters.")

    print("\n--- Password Strength Analysis ---")
    print("Length:", length)
    print("Complexity Score (0-4):", score)
    if feedback:
```



```python
        print("Suggestions:")
        for tip in feedback:
            print("-", tip)
    else:
        print("Strong password ✅")

# Function to generate custom wordlist
def generate_variations(words):
    variations = []
    for word in words:
        variations.append(word)
        variations.append(word + "123")
        variations.append(word + "!")
        variations.append(word.capitalize())
        variations.append(word[::-1])
        variations.append(word + "2024")
        variations.append(word + "@" + words[0])
        leet = word.replace("a", "@").replace("s", "$").replace("o", "0").replace("e", "3")
        variations.append(leet)
    return list(set(variations))  # remove duplicates

# Save the wordlist to a text file
def save_wordlist(words):
    with open("wordlist.txt", "w") as f:
        for word in words:
            f.write(word + "\n")
    print("\n--- Wordlist Generator ---")
    print("✅ Wordlist saved as 'wordlist.txt'")
```

5. Results

- Passwords were correctly evaluated for strength
- Wordlist file was successfully created with all combinations
- User-friendly and functional command-line tool was developed

6. Real-World Applications

- Ethical hacking and red teaming
- Penetration testing in cybersecurity assessments
- Simulating password cracking attempts using custom wordlists
- Training users to improve their password habits

7. Conclusion

This project effectively meets its goals by providing both password analysis and wordlist generation functionalities in a simple, CLI-based tool. It helps bridge the gap between user awareness and cybersecurity practice, making it a valuable learning and testing tool.