Password Strength Analyzer with Custom Wordlist Generator

Abstract

This project analyzes a given password for length, character variety, entropy, and common weaknesses such as repeats, sequences, and dictionary patterns. It outputs a score (0–10), a strength label (Weak/Medium/Strong), and human-readable advice. Additionally, it generates a custom wordlist using inputs like name, date of birth, pet name, and keywords. The generator applies leetspeak substitutions, combines tokens with separators, and appends or prepends year patterns. The output wordlist is exported as a .txt file for use with security testing tools. The solution is implemented in Python with a simple CLI interface and no external dependencies.

Introduction

Passwords remain the first line of defense for most systems. Weak or predictable passwords often result in unauthorized access and security breaches. This project implements a lightweight tool that evaluates password strength and generates targeted wordlists from user-related information. The goal is to demonstrate secure design principles and educate users about strong password creation, while also showing how custom wordlists are used in penetration testing and security assessments.

Tools Used :

- Python 3.8+

- Standard Python libraries: argparse, re, itertools, math, datetime

Steps involved in project :

1. Requirement Mapping – Followed the internship brief to include password analysis, custom wordlist generation, leetspeak substitutions, year patterns, and .txt export.

2. Design – Divided the project into two parts: (a) scoring engine (length, classes, entropy, pattern checks), and (b) wordlist generator (tokens, leetspeak, combinations, years).

3. Implementation – Built the script with three subcommands: analyze, generate, and both. Added password scoring logic, penalties for weak patterns, token normalization, and leetspeak mapping.

4. Testing – Verified sample inputs and ensured outputs respect size limits using --limit. Checked password scoring accuracy and generated wordlists for correctness.

5. Usage & Output – Provided CLI commands in the README. Outputs include password strength evaluation in the terminal and a .txt wordlist file.

Conclusion

The tool fulfills the internship deliverables by providing both a password strength evaluation and a flexible wordlist generator aligned with real-world attack patterns. It is lightweight, portable, and can run on any system with Python installed. The project demonstrates practical cybersecurity concepts and can be extended with a GUI or integrated with advanced libraries like zxcvbn for improved analysis.