Password Strength Analyzer with Custom Wordlist Generator

Abstract

The Password Strength Analyzer with Custom Wordlist Generator is a Python project that evaluates password strength and creates targeted wordlists for security testing. The tool supports two workflows: a command-line/password-analyzer mode that produces a full mywordlist.txt, and a GUI mode (Tkinter) that produces a limited custom_wordlist.txt for practical testing.

Introduction

Passwords protect digital assets, yet weak or guessable passwords remain a common vulnerability. This project provides both analysis and attack-oriented wordlist generation to help developers, testers, and security learners understand and test password resilience in realistic scenarios.

Tools Used

- **Python** — core implementation language
- **zxcvbn** — strength estimation (score, entropy, crack times, feedback)
- **Tkinter** — GUI for the interactive mode
- **argparse / CLI** — command-line interface for batch/automated runs
- **Custom wordlist generator** — combinatorics, leetspeak substitutions, appended years
- **File I/O** — export wordlists as .txt

Project Workflows (Two Modes)

1) Password Analyzer (CLI) → mywordlist.txt

- **Purpose:** Produce a comprehensive wordlist (full output) suitable for offline testing or feeding into cracking tools.
- **Inputs:** Password (or batch of passwords), and optional clues (names, dates, pet names) supplied via CLI arguments or a config file.
- **Processing:** generate_custom_wordlist() expands clues into permutations, leetspeak variations, and appended numbers/years to build an exhaustive list.
- **Output:** mywordlist.txt (full, unconstrained list). Useful for deep testing but may be large; use with caution.

2) GUI Password Analyzer (Tkinter) → custom_wordlist.txt (limited)

- **Purpose:** Provide a user-friendly interface that analyzes a single password and exports a practical-sized wordlist for targeted testing.
- **Inputs:** Single password entered into the GUI, plus comma-separated clues in the GUI field.
- **Processing:** Same generator is used, but the GUI limits the exported list to **500 entries** to avoid overwhelming users and files.
- **Output:** custom_wordlist.txt (first 500 entries) and, optionally, the full mywordlist.txt saved alongside it if requested.

Steps Involved in Building the Project

1. **Set up environment**: Install Python and required packages (zxcvbn, nltk if used, etc.).
2. **Implement analyzer**: analyze_password() wraps zxcvbn to extract score, crack-time estimates, warnings, and suggestions.
3. **Create generator**: generate_custom_wordlist(clues) builds permutations, leetspeak transformations, and appends numbers/years.
4. **CLI and GUI**: Implement argparse-based CLI to produce mywordlist.txt; implement Tkinter GUI that calls the same generator but saves a 500-entry custom_wordlist.txt (and optionally mywordlist.txt).
5. **File saving & UX**: Use file dialogs in GUI for save location; display analysis summary and number of entries written.
6. **Testing**: Verify both modes produce expected outputs and that the GUI truncation works reliably.

Conclusion

The project provides two complementary approaches for password analysis and wordlist generation - a comprehensive Command Line Interface (CLI) mode for in-depth testing and a Graphical User Interface (GUI) mode for quick, user-friendly assessments. Together, these modules create a powerful and flexible toolkit for security testing, education, and awareness.

The CLI mode supports advanced analysis with customizable options, making it suitable for penetration testers and researchers who require detailed control. The GUI mode simplifies usability, enabling beginners and professionals alike to evaluate password strength efficiently and export optimized wordlists. Overall, the project enhances understanding of password security, encourages strong password practices, and serves as a valuable learning resource for cybersecurity students and ethical hackers.