# Report on: Password Strength Analyzer & Wordlist Generator

Kotipalli Gautham Naga Ravi

November 13, 2025

## Introduction

In the current digital age, password security is a paramount concern. A significant vulnerability arises from users creating weak or predictable passwords, often based on easily accessible personal information. This project, `password_analyzer.py`, directly addresses this issue by providing an educational tool. It is a Python application designed to demonstrate *why* such passwords are insecure. The program operates in two modes: a Graphical User Interface (GUI) for ease of use and a Command-Line Interface (CLI) for flexibility. It provides users with two core functions: analyzing the strength of an existing password and generating a custom "attack" wordlist based on personal details, showing how easily such passwords can be guessed.

## Abstract

This report outlines a dual-mode Python application built to enhance password security awareness. The tool provides two primary functionalities: a password strength analyzer and a custom wordlist generator. The analyzer leverages the `zxcvbn` library to provide a quantitative score (0-4), an estimated crack time, and actionable feedback for any given password. The wordlist generator algorithmically creates a dictionary of potential passwords by taking personal inputs (like name, pet name, and birth year), applying common leetspeak substitutions (e.g., 'e' to '3'), and appending common numbers and years. This generated list highlights the vulnerabilities of using personal data in passwords. The application is implemented with both a user-friendly GUI using `tkinter` and a functional `argparse`-driven CLI, ensuring wide accessibility.

## Tools Used

The development of this project relied on several key Python libraries and modules:

- **Python 3:** The core programming language used for all logic.
- **Tkinter:** The standard Python library for creating the Graphical User Interface (GUI), including windows, labels, entry fields, and buttons.
- **zxcvbn library:** A third-party password strength estimator that provides the core analysis, scoring, and feedback.
- **itertools:** A standard Python module, specifically used for `combinations` to efficiently generate all possible leetspeak variants of a word.
- **argparse:** A standard Python module used to create the Command-Line Interface (CLI) and parse command-line arguments (i.e., the `--cli` flag).
- **sys:** A standard module used to interact with the Python runtime system.

## Steps Involved in Building the Project

The project was developed in a logical, phased approach:

1. **Core Logic - Strength Analysis:** The first step was to implement the password strength analysis. This involved integrating the `zxcvbn` library and creating the `analyze_password_strength` function to wrap its results in a simple, readable format.

2. **Core Logic - Wordlist Generation:** This was the most complex logic component. It involved:

   - Defining substitution rules (the `LEET_MAP`, `YEARS`, `NUMBERS`).
   - Creating the `leetspeak_variants` helper function using `itertools.combinations` to generate all variants of a single word.
   - Building the main `generate_wordlist` function to combine all user inputs, apply leetspeak, append numbers/years, and combine different inputs (e.g., name + year).

3. **GUI Development (Tkinter):** A `PasswordAnalyzerGUI` class was created to encapsulate the visual elements. This involved laying out the `Label` and `Entry` widgets for input, adding `Button` widgets, and creating handler functions (`check_strength` and `generate_wordlist`) to link the GUI to the core logic. A `Text` widget was used for displaying feedback.

4. **CLI Development (argparse):** To provide an alternative to the GUI, the `run_cli_mode` function was written. This function uses simple `input()` calls to gather data and `print()` calls to display results.

5. **Integration and Main Entry Point:** Finally, the `main()` function was created. It uses `argparse` to check if the `--cli` flag was provided by the user. Based on this, it either launches the CLI mode or initializes and runs the Tkinter GUI loop.

## Conclusion

The ”Password Strength Analyzer & Wordlist Generator” project was successfully implemented as a functional and educational security tool. It effectively demonstrates the significant risks associated with creating passwords based on personal, guessable information. By providing both a strength-checking mechanism and a wordlist generator, it serves as a practical lesson in password hygiene. Potential future enhancements could include integrating an API to check against known password breaches (like Pwned Passwords) or expanding the wordlist generation rules for even more comprehensive (and realistic) attack dictionaries.