Password Strength Analyzer and Custom Wordlist Generator: Project Report

Introduction:
In modern digital security, the human element remains the weakest link, often manifesting through the creation of predictable and weak passwords. An attacker's initial step, often termed "password profiling," involves generating highly targeted wordlists based on known personal information (such as names, dates, or pets). This project addresses this vulnerability by building a dual-purpose Python application: a Password Strength Analyzer and a Custom Wordlist Generator. The tool provides users with actionable security feedback and demonstrates, through its wordlist generation capabilities, the exact permutations an attacker might use, fostering greater awareness of password risk.

Abstract:
This report details the development of a security utility designed to enhance password assessment and support penetration testing efforts. The application provides a user-friendly Graphical User Interface (GUI) built with Tkinter, offering two core functions. First, it utilizes the advanced zxcvbn library for realistic password strength analysis, providing a 0-4 security score and actionable suggestions. Second, it implements a highly customizable wordlist generator. This generator accepts personal context inputs (name, date, pet) and applies mutation rules, including leetspeak and year appending, to create a targeted dictionary file for export. The resulting tool is a self-contained, interactive utility for security enthusiasts and developers.

Tools Used:

Tool/Library	Role in Project
Python 3.x	The core programming language used for all logic and functions.
Tkinter	Python's standard library for creating the cross-platform Graphical User Interface (GUI), providing user interaction via two main tabs.
zxcvbn	A realistic password strength estimator. It uses pattern matching and entropy calculations to provide a security score (0-4) and helpful feedback, accurately detecting common passwords and patterns.
itertools	Used specifically for the product function to efficiently generate all possible permutations of leetspeak substitutions for base words.
datetime	Used to parse and manipulate date strings (YYYY-MM-DD) and

	dynamically generate a range of recent years for appending to the wordlist.

Steps Involved in Building the Project:

The project followed a structured development approach, integrating security libraries with a user-friendly interface.

1. Project Setup and Core Logic Definition
The initial step involved defining the two distinct functional modules: analyze_password_logic and generate_wordlist_logic. A global LEET_MAP dictionary was created to define substitution rules (e.g., 'a': ['4', '@']), which is crucial for the wordlist generation.

2. Password Analysis Implementation
The analyze_password_logic function was built around the imported zxcvbn function. This function accepts the target password and an optional list of "user inputs" (context). The results, including the numerical score, estimated crack time (crack_times_display), and suggestions, were parsed and formatted into a human-readable string for display in the GUI.

3. Custom Wordlist Generation Logic
The generate_wordlist_logic function was the most complex component, focusing on mutations and permutations:

- Base Word Collection: Inputs (name, pet) were collected and converted to lowercase, capitalized, and uppercase versions to form a base set.
- Leetspeak Permutation: The itertools.product function was used on the base words and the LEET_MAP to generate every possible leetspeak combination efficiently.
- Date and Year Append: Date strings were parsed into common formats (YY, MMDD, YYYYMMDD). A range of years (configurable via year_range) was generated. These years and date parts were then appended to all base words and leetspeak variants.
- Export: The final set of unique words was sorted and written line-by-line to the specified .txt output file.

4. Tkinter GUI Development
The CLI version was replaced with a Tkinter application, PasswordToolGUI, which encapsulates the entire user experience.

- Notebook Structure: A ttk.Notebook was used to create two tabs: "Password Analysis" and "Wordlist Generation," improving navigation.
- Input Handling: ttk.Entry widgets were created for all inputs (password, name, date) and a ttk.Spinbox for the year range. Password entry visibility was controlled using a ttk.Checkbutton.
- File I/O Integration: The filedialog module was used to provide a Browse button, allowing users to visually select the output file path, replacing manual path entry.
- Output Display: A tk.Text widget was implemented in both tabs to display status messages and detailed analysis reports.

Conclusion:
The project successfully delivered a robust and user-friendly security tool. By integrating the cutting-edge analysis of zxcvbn with highly targeted wordlist generation capabilities, the application meets all project objectives. It is valuable not only as a penetration testing utility but also as an educational tool to visually demonstrate how easily personal data translates into high-probability guesses. Future work could involve expanding the mutation dictionary to include common separators (-, !) and adding capitalization rules beyond simple case conversions for enhanced wordlist complexity.