# Project Report: Password Strength Analyzer with Custom Wordlist Generator

June 26, 2025

## 1 Introduction

The Password Strength Analyzer with Custom Wordlist Generator is a cybersecurity tool designed to evaluate password strength and generate tailored wordlists for penetration testing. This project addresses the need for robust password security by analyzing passwords against industry-standard metrics and creating custom wordlists to simulate potential attack vectors. The tool is intended for ethical hacking and educational purposes, enabling users to assess password vulnerabilities and generate attack-specific wordlists compatible with cracking tools like Hashcat and John the Ripper.

## 2 Abstract

This project delivers a Python-based tool that combines password strength analysis with custom wordlist generation. The tool uses the `zxcvbn` library for password strength evaluation, computing metrics such as strength score, estimated crack time, and entropy. It also generates wordlists based on user inputs (name, date of birth, pet name, password) with variations including leetspeak substitutions and appended years. The application supports both a graphical user interface (GUI) built with `tkinter` and a command-line interface (CLI) using `argparse`, ensuring flexibility for different user needs. Wordlists are exported as `.txt` files, optimized for use in password cracking tools. The project emphasizes security, usability, and compatibility with ethical hacking workflows.

## 3 Tools Used

- **Python**: Core programming language for implementing the tool's logic.
- **zxcvbn**: Library for robust password strength analysis, providing score, crack time, and feedback.
- **NLTK**: Used to enhance wordlists by incorporating related English words from the `words` corpus.
- **argparse**: Enables CLI support for automated password analysis and wordlist generation.
- **tkinter**: Provides a user-friendly GUI for interactive password analysis and wordlist creation.

## 4 Steps Involved in Building the Project

1. **Setup and Dependency Installation**: Installed required Python libraries (`zxcvbn`, `nltk`) and ensured the NLTK `words` corpus was downloaded for wordlist generation.

2. **Password Strength Analysis Implementation**: Integrated `zxcvbn` to compute strength scores (0–4), estimated crack times, and user feedback. Added a custom entropy calculation based on character set size (lowercase, uppercase, digits, special characters) to complement `zxcvbn`'s metrics.

3. **Wordlist Generation Logic**: Developed functions to generate wordlists from user inputs (name, date of birth, pet name, password). Implemented leetspeak substitutions (e.g., 'a' → '4', '@') and year appending (1900–2025). Used NLTK to include related words based on substring matching.

4. **Interface Development**: Built a `tkinter`-based GUI with input fields for password, name, date of birth, and pet name, along with checkboxes for leetspeak and year options. Implemented a CLI using `argparse` for command-line execution with arguments for password and wordlist parameters.

5. **Output and Export Functionality**: Designed the tool to display analysis results (strength, crack time, entropy, suggestions) in both GUI and CLI. Enabled wordlist export as `.txt` files, ensuring compatibility with cracking tools.

6. **Testing and Validation**: Tested password analysis with various inputs (e.g., "123", "MyPass123", "p@ssw0rd!") to verify `zxcvbn` accuracy. Validated wordlist generation with sample inputs (name="John", dob="1990-01-01", pet="Max"), confirming thousands of variations. Ensured `.txt` exports worked with Hashcat.

## 5 Conclusion

The Password Strength Analyzer with Custom Wordlist Generator successfully meets its objectives of evaluating password strength and generating attack-specific wordlists. The integration of `zxcvbn` ensures accurate and industry-standard password analysis, while the custom wordlist generator enhances penetration testing capabilities with leetspeak and year-based variations. The dual GUI and CLI interfaces provide flexibility for both novice and advanced users. The tool's compatibility with cracking tools and its focus on ethical use make it valuable for cybersecurity education and testing. Future enhancements could include regex-based pattern matching, progress indicators for large wordlists, and support for additional output formats, further improving its utility in real-world scenarios.