**Password Strength Analyzer with Custom Wordlist Generator:**

**Introduction**

Passwords remain the most widely used form of authentication, but weak or predictable passwords are a major security risk. Attackers often exploit human tendencies, such as using names, dates, or simple patterns. This project develops a tool that evaluates password strength and generates customized wordlists for penetration testing and security research. By combining analytical libraries with user-specific data, the tool highlights vulnerabilities and demonstrates how easily weak credentials can be guessed.

**Abstract**

The Password Strength Analyzer integrates password evaluation and wordlist creation in a single tool. Using **zxcvbn**, a popular password strength estimator, or entropy-based calculations, it scores user passwords against dictionary and pattern attacks. In addition, the tool accepts user-defined inputs such as names, dates, or pet names to generate personalized wordlists. These lists include transformations like leetspeak (e.g., replacing letters with numbers) and common suffixes (e.g., adding years). The final wordlists are exported in **.txt** format, suitable for security testing with cracking tools.

**Tools Used**

* **Python** – Core programming language.
* **argparse** – Command-line argument parsing.
* **NLTK** – For text handling and tokenization.
* **zxcvbn** – Password strength analysis.
* **tkinter** – Optional GUI interface.

**Steps Involved in Building the Project**

1. **Password Strength Analysis**
   * Input a password through CLI or GUI.
   * Use zxcvbn to calculate strength, entropy, and estimated crack time.
   * Display results to the user.
2. **Custom Input Collection**
   * Collect user-related words: name, birthday, pet, etc.
   * Normalize inputs using NLTK.
3. **Wordlist Generation**
   * Apply transformations:
     + Leetspeak substitutions (e.g., a → @, e → 3).
     + Common suffixes (e.g., 123, 2024, !, ?).
     + Case variations (lower, upper, camel case).
4. **Export Wordlist**
   * Save generated variations into a .txt file.
   * Ensure compatibility with password-cracking tools (e.g., John the Ripper, Hashcat).
5. **User Interface**
   * Provide CLI with argparse for automation.
   * Optional tkinter GUI for non-technical users.

**Conclusion**

This project emphasizes the importance of strong, unpredictable passwords. By simulating real-world attack strategies with custom wordlists, users can understand how personal data contributes to password weakness. The analyzer and generator serve both educational and security testing purposes, helping organizations and individuals improve password hygiene. Future improvements could include integration with online breach databases to warn users if their chosen passwords have appeared in leaks.