**Password Strength Analyzer with Custom Wordlist Generator**

* **Introduction:**

In today’s digital world, securing user accounts with strong passwords is critical. Weak passwords are vulnerable to attacks like brute force and dictionary attacks. This project focuses on building a **Password Strength Analyzer** that evaluates the strength of user-created passwords and a **Custom Wordlist Generator** that creates targeted wordlists for ethical hacking or penetration testing. This tool helps in both improving password security and generating test data for password cracking scenarios.

* **Abstract:**

The main objective of this project is to design a dual-purpose tool that:

* **Analyzes Password Strength:**  
  Checks password length, complexity, and entropy. Gives feedback (Weak, Moderate, Strong) based on parameters like uppercase, lowercase, digits, symbols, and overall randomness.
* **Generates Custom Wordlists:**  
  Creates wordlists based on user-defined patterns, keywords, and rules. This is useful for testing password strength using dictionary attacks during cybersecurity assessments.

This project helps both end-users (to create strong passwords) and cybersecurity professionals (to generate targeted attack wordlists for penetration testing in a controlled environment).

* **Tools Used:**

| **Category** | **Tool** | **Purpose** |
| --- | --- | --- |
| **Programming Language** | Python (Pre-installed) | For coding the analyzer and generator. |
| **Wordlist Tools** | **Crunch** | To generate custom wordlists with specific patterns and length. |
| **Password Cracking Tools (for testing wordlists)** | **John the Ripper**, **Hydra**, **Medusa** | To test the effectiveness of generated wordlists by attempting password cracking (for ethical testing only). |
| **Network Testing/Brute Forcing** | Hydra | For testing generated wordlists against online services like SSH, FTP. |
| **Dictionary Files** | **RockYou.txt**, **SecLists**, **Common.txt** | Pre-built wordlists in Kali for reference or merging with custom wordlists. |
| **Text Editors** | Nano, Vim, or Leafpad | For quickly editing wordlists or scripts. |
| **Python Libraries** | re, string, random, itertools (all installable via pip if missing) | For handling string manipulation, regex, and wordlist logic. |
| **Terminal / Bash Shell** | Terminal (Default in Kali) | To run Python scripts and other command-line tools. |

* **Steps Involved in Building the Project:**

1. **Password Strength Analyzer Module:**
   * Input password from user.
   * Check for:
     + Password length.
     + Use of uppercase letters.
     + Use of lowercase letters.
     + Inclusion of digits.
     + Special characters.
   * Calculate an overall **strength score**.
   * Provide user feedback:  
     *(Example: "Weak", "Moderate", "Strong", with suggestions to improve).*
2. **Custom Wordlist Generator Module:**
   * Take user input for:
     + Keywords (e.g., name, birth year, favorite color).
     + Minimum and maximum length of words.
     + Patterns (e.g., adding numbers, special symbols).
   * Generate combinations like:
     + Simple concatenation.
     + Leetspeak substitutions (e.g., a → @, s → $).
     + Common password patterns (e.g., Name123, Name@2025).
   * Save the generated wordlist into a .txt file.
3. **Testing and Validation:**
   * Test password analyzer with different passwords.
   * Generate sample wordlists with various settings.
   * Run simple dictionary attacks using tools like Hydra or John the Ripper (only for educational/testing purposes).

* **Conclusion:**

This project provided hands-on experience with Python programming, regular expressions, string manipulation, and file handling. It also strengthened understanding of **password security principles**, **cybersecurity testing**, and **ethical hacking techniques**.

The Password Strength Analyzer helps users create stronger passwords, while the Custom Wordlist Generator aids penetration testers in simulating real-world password attacks in ethical testing environments.