Password Strength Analyzer & Custom Wordlist Generator

Introduction

- **Purpose**: Evaluate password strength and generate targeted wordlists to demonstrate real-world risk.
- **Scope**: A lightweight Python CLI with three modules — Analyze, Audit, and Wordlist Generator.
- **Deliverables**: `psawg.py`, `README.md`, `report.pdf`, `wordlist.txt`, sample data & screenshots.

Abstract

- **Analyze**: Computes Shannon entropy and uses ZXCVBN for realistic scoring, feedback, and crack-time estimates.
- **Audit**: Processes password lists and exports structured CSV reports for bulk assessment.
- **Wordlist Generator**: Produces targeted wordlists from user metadata (case variants, separators, suffixes, years) with size limits.
- **Goal**: Provide defensive insights and awareness by showing how attackers may craft targeted guesses.

Tools Used

- Python 3.x (virtual environment)
- Libraries: "**zxcvbn**", "**tqdm**", **argparse**"
- Development: Git & GitHub
- Environment: Kali Linux / Windows (any Python-capable OS)
- Utilities: CSV output, simple CLI (argparse), and basic file I/O

Steps Involved in Building the Project

1. **Environment setup:**
   - Create and activate Python virtual environment (`python3 -m venv .venv`).
   - Install dependencies: `pip install zxcvbn tqdm`.

2. **Analyzer implementation**
   - Implement Shannon entropy calculation and integrate ZXCVBN for realistic scoring and feedback.
   - Produce JSON output with fields: length, entropy_bits, zxcvbn_score, crack-time, feedback, class.

3. **Audit module**
   - Read password list from file and run analyzer on each entry.
   - Export results to `reports/analysis.csv` with columns: password, length, entropy_bits, zxcvbn_score, crack_time, class.

4. **Wordlist generator**
   - Load metadata (CLI args or JSON).
   - Generate variants (case, title, upper), leetspeak substitutions, separators, common suffixes and year patterns.
   - Combine words (single & two-word combinations) and cap output size with `--max` parameter.

5. **Testing & validation**
   - Run example commands and capture screenshots for analyze, audit, and wordlist outputs.
   - Verify `wordlist.txt` size with `wc -l` and validate CSV entries with `head`.

6. Documentation & submission

- Write README with usage examples and embed screenshots.
- Add `report.pdf` and push the repository to GitHub for submission.

Conclusion

- The tool effectively identifies weak passwords and provides actionable feedback to improve password hygiene.
- Custom wordlists demonstrate how attacker-controlled metadata can significantly reduce cracking time.
- The project is ready for submission: code, documentation, screenshots, and a concise report are available in the GitHub repository.