Project Report: PDF Cracker Tool

1. Objective

The goal of this project was to develop a command-line-based tool to simulate brute-force password attacks on **encrypted PDF files**. The project aimed to highlight the security weaknesses of poorly protected documents and demonstrate how attackers or security analysts might test the strength of document-level encryption in real-world scenarios.

2. Project Summary

This project involved building a Python-based tool that performs **dictionary attacks** on password-protected PDF files. The tool accepts a user-defined wordlist and attempts to open the file with each password sequentially. If the correct password is found, the tool prints it to the terminal.

The simulation represents the methodology used by both attackers and digital forensic analysts in attempting to gain access to secured documents without prior knowledge of the password. The tool was implemented with optional **multithreading support** for performance enhancement and features robust error handling.

All testing was conducted on self-created password-protected PDF files within a secure, offline environment.

3. Key Functionalities

- **Dictionary-based Brute Force**: Accepts a wordlist and tries each password against the PDF until it finds a match.
- **Encrypted PDF Support**: Works with PDFs encrypted using standard user/owner password protection.
- **Command-Line Interface (CLI)**: Built using Python's `argparse` for flexible usage.
- **Optional Multithreading**: Can be configured to speed up brute-force attempts using concurrent threads.
- **Error Resilience**: Handles incorrect or corrupted PDF files gracefully and logs failed attempts.

4. Tools and Technologies Used

- **Python 3.x**: Core development language
- `pikepdf` / `PyPDF2`: PDF decryption and validation
- `argparse`: Command-line argument handling
- `threading` **(optional)**: For concurrent password attempts
- **Sample PDF and Password Lists**: Created internally for testing purposes

5. Workflow Overview

1. The user provides a path to a password-protected PDF file and a text file containing possible passwords.
2. The tool reads each password and attempts to decrypt the PDF.
3. If decryption is successful, the correct password is printed and the attack stops.
4. All operations are logged, and invalid/corrupted attempts are skipped to maintain stability.

6. Ethical Considerations

This project was developed and executed in a **controlled lab environment** using sample documents created by the developer. It was not used on third-party or unauthorized PDF files.

The purpose was strictly **educational** — to understand how brute-force password attacks work and how weak document encryption can be exploited. It also serves as a training tool for blue teams to test document security policies and forensics professionals in recovery tasks.

7. Learning Outcomes

- Understood the inner workings of **PDF encryption mechanisms**.
- Gained hands-on experience in **password cracking techniques** and **dictionary attacks**.
- Learned how to implement **robust CLI utilities** and handle real-world edge cases in file decryption.
- Practiced building multi-threaded tools in Python for performance-critical tasks.
- Developed a stronger understanding of how document security can be bypassed through brute force when weak passwords are used.

8. Conclusion

The PDF Cracker Tool project provided valuable insights into the **vulnerabilities of encrypted documents** when protected by weak or common passwords. It replicated an essential step in offensive security and forensics: recovering access to locked files.

The project also reinforced the importance of using strong, unpredictable passwords and illustrated the ease with which documents can be compromised when security best practices are ignored.