PDF Password Cracker Tool

Domain: Cybersecurity | **Type:** Internship Project

A Python-based tool designed to test password strength of encrypted PDF files in controlled environments. Demonstrates the critical importance of strong passwords through authorized security testing and educational exploration.

Author : Akash Motghare

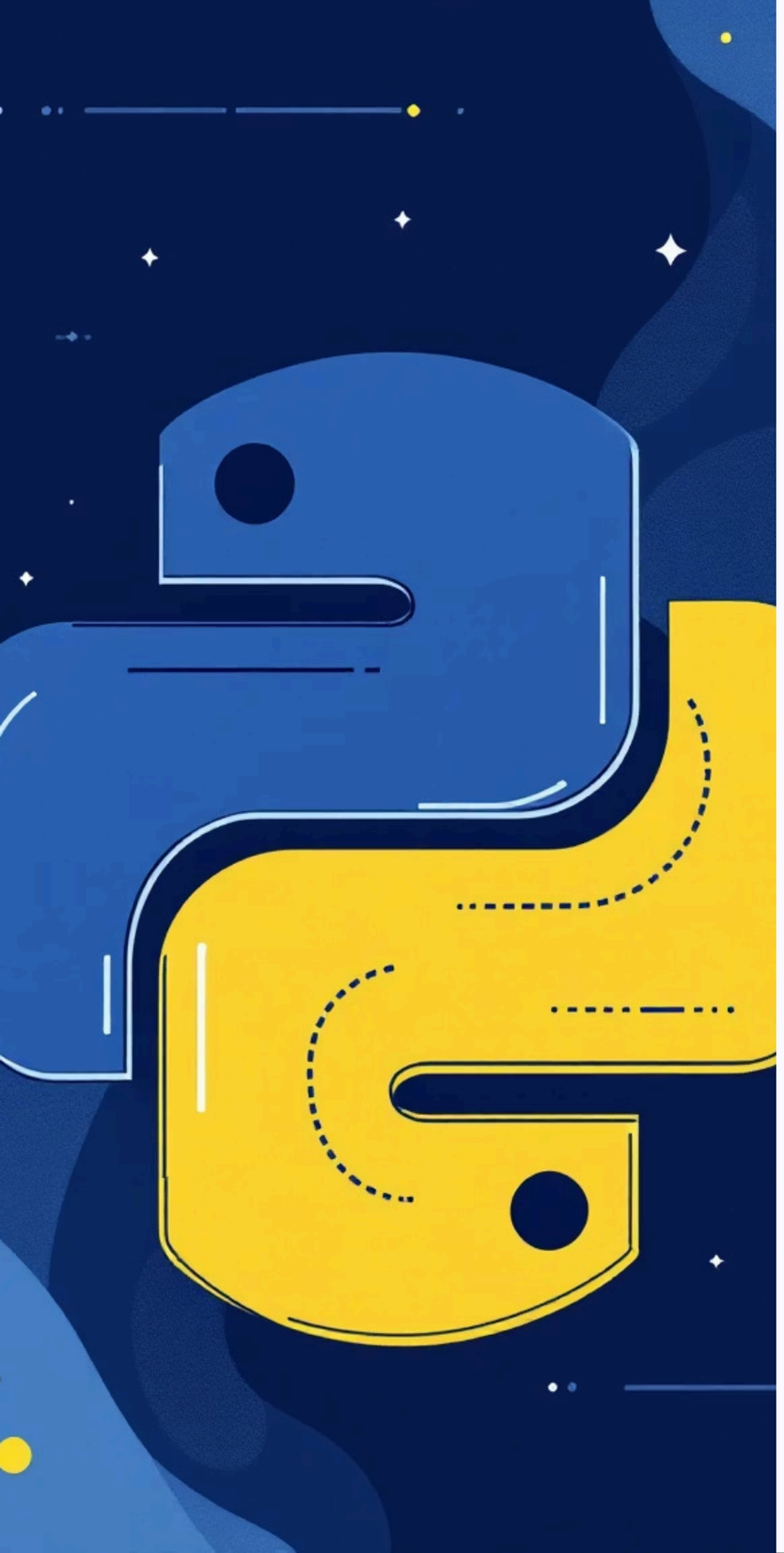
Problem Statement & Objectives

The Challenge

- Password-protected PDFs depend entirely on password strength for security
- Weak or commonly used passwords significantly reduce encryption effectiveness
- Manual testing of password security proves inefficient and time-consuming
- Organizations need tools to validate their document security posture

Project Objectives

- Implement both dictionary-based and brute-force password testing methods
- Utilize multithreading techniques to improve performance and efficiency
- Demonstrate real-world password attack concepts in a controlled setting
- Provide practical insights into PDF encryption vulnerabilities



Technical Architecture

Core Technologies

- Python 3 programming language
- pikepdf library for PDF manipulation
- argparse for command-line interface
- concurrent.futures for parallel processing
- tqdm for progress visualization

Attack Methodology

- Load and analyze encrypted PDF file
- Select attack method based on scenario
- Execute concurrent password attempts
- Terminate upon successful recovery
- Report results and performance metrics

Implementation Features

01

Command-Line Interface

Flexible argument parsing system supporting multiple attack modes and configuration options for different testing scenarios.

02

Dictionary Attack Mode

Leverages pre-compiled wordlist files containing common passwords, enabling rapid testing against known weak credentials.

03

Brute-Force Attack Mode

Configurable character set and length parameters allow systematic testing of all possible password combinations.

04

Concurrent Execution

Multithreading implementation distributes password attempts across multiple CPU cores for optimal performance.

05

Smart Termination

Automatic execution halt upon password recovery prevents unnecessary computational resource consumption.

Project Screenshot

```
PS F:\Projects\Pdf-cracker-tool> python .\main.py -h
usage: main.py [-h] [-w WORDLIST] [-b] [-min MINLEN] [-max MAXLEN] [-c CHARSET] [-t THREADS] pdf

PDF Password Cracker Tool

positional arguments:
  pdf                   Path to the password-protected PDF file

options:
  -h, --help            show this help message and exit
  -w WORDLIST, --wordlist WORDLIST
                        Path to wordlist file (optional)
  -b, --brute           Enable brute-force attack
  -min MINLEN, --minlen MINLEN
                        Minimum password length
  -max MAXLEN, --maxlen MAXLEN
                        Maximum password length
  -c CHARSET, --charset CHARSET
                        Character set for brute-force
  -t THREADS, --threads THREADS
                        Number of threads
PS F:\Projects\Pdf-cracker-tool> python .\main.py -min 6 F:\Projects\pdf-protection-tool\protected_Port-Scanner-Using-Python.pdf -b -c abcdef -max 6
Brute-forcing: 46656it [00:00, 115357.63it/s]

[+] Password found: abcdef
PS F:\Projects\Pdf-cracker-tool>
```

Results, Limitations & Ethical Considerations

Key Results

- Successfully recovered weak passwords in dictionary attacks within seconds
- Brute-force method proved effective for short passwords under 6 characters
- Multithreading reduced execution time by up to 75 percent compared to single-threaded approach
- Tool demonstrated clear correlation between password complexity and cracking difficulty

Technical Limitations

- Ineffective against strong passwords with high entropy and sufficient length
- High computational cost for brute-force attacks on passwords exceeding 8 characters
- Performance constrained by available CPU resources and memory
- Dictionary attacks limited by wordlist quality and comprehensiveness

Ethical Guidelines

- Use exclusively on personally owned or explicitly authorized PDF files
- Never deploy for unauthorized access or malicious purposes
- Respect all applicable computer security laws and regulations
- Maintain responsible disclosure practices for discovered vulnerabilities

Learning Outcomes & Conclusion

Key Learning Outcomes

- Deep understanding of PDF encryption mechanisms and security protocols
- Practical exposure to dictionary and brute-force password attack techniques
- Hands-on experience implementing multithreading in Python for performance optimization
- Critical awareness of password security best practices and common vulnerabilities
- Knowledge of responsible security testing methodology and ethical boundaries

Project Conclusion

This project demonstrates that weak passwords fundamentally compromise document security regardless of encryption strength. Organizations must enforce strong password policies and user education programs.

Strong passwords combined with robust encryption remain essential for protecting sensitive information in PDF documents and other digital assets.