Project Report:

Password Master OFF3NC3

Project Identifier: PM-OFF3NC3-V1.0
Developer: Ayushman Das

1. Executive Summary: The Need for Targeted Wordlists

1.1 Project Mandate and Offensive Security Objective

The **Password Master OFF3NC3 (PM-OFF3NC3)** project was developed in response to the ethical security testing community's need for highly targeted, customized wordlists. Generic dictionary attacks are inefficient; modern threat modeling requires generating password candidates specifically tailored to a target individual or organization—a technique known as **"Password Profiling"** or **"OSINT-Driven Wordlist Generation."**

The primary objective was to engineer a tool that automates the complex combination and transformation of Personal Identifiable Information (PII) and common attack patterns, outputting a high-quality, formatted .txt wordlist suitable for immediate use with security testing platforms like Hashcat or John the Ripper.

1.2 Architectural and Methodological Overview

The PM-OFF3NC3 is built on a specialized **Generation-and-Export Pipeline** designed for algorithmic complexity and reliable file handling.

Component	**Technology**	**Role in Offensive Testing**
Generation Engine	Python Functions & Dictionaries	Implements the complex leetspeak and year-appending logic to maximize password candidate permutations.
Tool Interface	Python (tkinter)	Provides a secure,

		dedicated UI for analysts to input PII (Name, Pet, Year) used as base words for the attack dictionary.
Export Module	Python (os, filedialog)	Ensures reliable file creation (.txt format) and proper line-by-line formatting necessary for cracking tools.

2. Algorithmic Design and Data Preprocessing:

2.1 PII Input Strategy

The tool focuses on capturing low-hanging PII commonly used in weak passwords:

1. **Name/Alias:** Primary anchor for personal passwords.
2. **Pet's Name:** A common emotional anchor, frequently used directly or with simple modifications.
3. **Custom Year/Date:** Used to target birth years, anniversaries, or default years (e.g., 2024).

2.2 The Core Generation Algorithm

The generator's strength lies in its ability to combine permutation techniques:

1. **Base Word Variation:** Automatically generates lowercase, capitalized, and raw combined strings (e.g., [Name][Pet], [Pet][Name]).
2. **Leetspeak Substitution:** A comprehensive LEET_MAP dictionary transforms common letters (a, e, s, t, o, i, g) into their numerical equivalents (4, 3, 5, 7, 0, 1, 9). This single substitution exponentially increases the wordlist's coverage of common user evasion techniques.
3. **Suffix Appending:** The system appends a predefined, highly optimized list of COMMON_YEARS and their 2-digit abbreviations (2024 -> 24) to every base word and every leetspeak variation.

This multi-level permutation strategy ensures that even simple input (e.g., "Max") can generate dozens of high-probability target candidates (e.g., Max1999, m4x24, m4x123).

3. Implementation and Engineering Challenges

3.1 Challenge 1: Ensuring Export Integrity for Cracking Tools

Description: Password cracking tools (like Hashcat) are highly sensitive to file formatting. Incorrect line breaks, encoding, or trailing spaces can render an entire wordlist useless.

Solution: The save_wordlist_to_file function was implemented using the filedialog module for secure file path selection. The critical steps were:

- Using with open(..., 'w', encoding='utf-8') to guarantee universal compatibility.
- Enforcing a consistent output by using a set() during generation to eliminate duplicates, followed by sorted(list(wordlist)) before writing, ensuring an ordered, unique, and efficient wordlist.
- Explicitly appending '\n' to each entry to guarantee the correct single-word-per-line structure.

3.2 Challenge 2: UI for Input and File Handling

The UI was designed to prioritize clarity and ease of export for the security analyst:

- **Dark Theme:** Maintains the professional, low-light environment required by analysts.
- **Clear Labeling:** Explicitly directs the user to input the three crucial PII components.
- **Generate & Export Button:** Combines the algorithmic complexity (generation) and the file system interaction (save dialog) into a single, intuitive action.

4. Security Auditing and Tool Validation

4.1 Penetration Testing Methodology

The tool's efficacy was tested by analyzing its output quality. A single test input, "Jessica," generated 10 unique base words (e.g., Jessica, jessica, j3551c4) which, when combined with the 14 available year suffixes, resulted in a highly targeted dictionary containing over **140 unique, high-probability password candidates**.

This demonstrates PM-OFF3NC3's superior efficiency compared to running general dictionary attacks.

4.2 Ethical Use Warning

Due to the powerful and specific nature of this profiling tool, a prominent **WARNING** label was integrated into the GUI, explicitly advising that the tool must be used *only* for ethical security testing and authorized auditing purposes.

5. Conclusion and Future Development

The **Password Master OFF3NC3** project successfully delivers a specialized tool for ethical security research, moving beyond simple static wordlists to algorithmic, PII-driven generation. The successful integration of leetspeak and structured appending demonstrates robust algorithmic development.

Future Scope:

1. **Rule-Based Generation:** Integrate the ability to export a base wordlist compatible with advanced rule engines (like Hashcat's Rule-Based Attack) for even deeper permutations (e.g., pre-pending symbols).
2. **Character Replacement:** Expand the LEET_MAP dictionary to include complex phonetic replacements and character doubling.
3. **Progress Tracking:** Implement a GUI element to display the real-time word count during generation, offering feedback on the complexity and size of the output dictionary.