

PASSWORD CRACKING TOOL

A project Report submitted to the Bharathiar University in Partial
Fulfillment of the requirement for the Award of the Degree of

MASTER OF SCIENCE IN CYBER SECURITY

Submitted By

PRATHIVRAJ A (23CSEH25)

Under the Guidance of

Dr. T. AMUDHA, MCA., M.Phil., Ph.D.,

Professor, Department of Computer Applications.



**DEPARTMENT OF COMPUTER APPLICATION
BHARATHIAR UNIVERSITY
COIMBATORE**

DECEMBER-2024

DECLARATION

I hereby declare that this mini project, submitted to the Department of Computer Applications, Bharathiar University, is a record of original work done by **PRATHIVRAJ A (23CSEH25)** under the supervision and guidance of **Dr. T. AMUDHA, MCA., M.Phil., Ph.D.,** Department of Computer Applications, Bharathiar University. I affirm that this mini project has not been submitted for the award of any Degree, Diploma, Associateship, Fellowship, or any similar title to any candidate at any other University.

Place: Coimbatore

Date:

Signature of the candidate
(PRATHIVRAJ A)

Countersigned by,

CERTIFICATE

This is to certify that the mini project titled “Password Cracking Tool”, submitted to the Department of Computer Applications, Bharathiar University in partial fulfilment of the requirements for the award of the degree of Masters of Science in Cyber Security, is a record of original work done by **PRATHIVRAJ A (23CSEH25)** under the supervision and guidance of **Dr. T. AMUDHA, MCA., M.Phil., Ph.D.**, during the period of study in the Department of Computer Applications, Bharathiar University, Coimbatore, and that this project work has not formed the basis for the award of any Degree/Diploma /Associateship/ Fellowship or similar title to any candidate of any University.

Place: Coimbatore

Date:

Project Guide

Head of the Department

Submitted for the Project VIVA-VOCE Examination held on

Internal Examiner

External Examiner

TABLE OF CONTENTS

Chapters	Topics	Page. No
Chapter 1	INTRODUCTION	3
	1.1 Objective	4
	1.2 Key features	5
Chapter 2	GUI DESIGN AND STRUCTURE	11
	2.1 Main Page	11
	2.2 Button Functionality	12
Chapter 3	WORKFLOW AND SYSTEM OVERVIEW	23
	3.1 Workflow	23
	3.2 System Overview	24
Chapter 4	DATA COLLECTION METHODOLOGIES	26
	4.1 Password Hah Collection	26
	4.2 Wordlist Collection	27
	4.3 Password Strength Data Collection	27
	4.4 Brute Force Data Collection	27
	4.5 Data Validation and Security	28
Chapter 5	CODE ARCHITECTURE AND IMPLEMENTATION	29
	5.1 Class-Based Architecture	29
	5.2 Data Handling	31
	5.3 Error Handling	32
	5.4 Code Optimization	33
	5.5 GUI Integration	33
Chapter 6	RESULTS	34
	6.1 Result of Dictionary Attack	34
	6.2 Result of Brute Force Attack	35
	6.3 Result of Wordlist Generator	36
	6.4 Result of Password Generator	37
	6.5 Result of Password Analyzer	38
Chapter 7	CONCLUSION	39
	REFERENCES	41
	Appendix A: List of figures.	42
	Appendix B: Sample code.	43

ABSTRACT

This project introduces a comprehensive Password Cracking Tool designed using Python's Tkinter library to simulate, analyze, and understand various aspects of password security and cracking techniques. With the increasing reliance on digital systems, ensuring robust password protection has become critical, and ethical tools like this play a vital role in understanding vulnerabilities and developing stronger defense mechanisms. The tool integrates multiple functionalities, including dictionary and brute force attacks, password generation, wordlist creation, and password strength analysis. The dictionary attack allows testing a hash against a user-specified dictionary file, while the brute force attack explores all possible combinations of characters to crack a target hash, demonstrating the feasibility and challenges of these approaches. It also includes a password generator for creating strong, randomized passwords and a wordlist generator for producing custom wordlists, catering to penetration testing scenarios. Furthermore, the password analyzer evaluates the strength of user-provided passwords based on complexity and character variety, offering insights into creating robust credentials. The user-friendly interface supports threading for responsiveness during intensive operations and provides features like progress bars and abort buttons for enhanced usability. By supporting modern hash functions such as SHA3-256 and BLAKE2b, the tool reflects real-world security practices. This application aims to educate users about the vulnerabilities of weak passwords and promote awareness of robust authentication strategies, serving as both a learning tool for cybersecurity enthusiasts and a foundational step toward ethical hacking practices. This project emphasizes the importance of password security and fosters a proactive approach to mitigating cybersecurity threats.

CHAPTER 1

INTRODUCTION

In today's digital era, the importance of secure password practices and the need to understand password vulnerabilities are more critical than ever. Passwords serve as the first line of defence against unauthorized access to personal and organizational systems. However, weak or poorly managed passwords remain a common vulnerability exploited by attackers. This project, a Password Cracking Tool, aims to provide a comprehensive understanding of password security by simulating real-world password-cracking scenarios in a controlled and educational environment. The Password Cracking Tool is designed to analyse and demonstrate the effectiveness of various password security measures while emphasizing the significance of creating strong and resilient passwords. Built using Python, the tool features a user-friendly Graphical User Interface (GUI) developed with Tkinter, ensuring accessibility for users of all technical backgrounds. Its capabilities include support for popular hashing algorithms such as MD5, SHA-256, SHA-3, and BLAKE2, along with the simulation of common attack methods like dictionary attacks and brute force attacks.

In addition to its password-cracking capabilities, the tool includes a password strength analyser, helping users evaluate the strength of their passwords and providing recommendations for improvement. Key features such as custom wordlist generation, multithreading for responsiveness, and error handling mechanisms enhance the tool's practicality and reliability. This documentation covers the objectives, design, and functionality of the Password Cracking Tool, detailing its key features, code structure, and implementation. It also provides a step-by-step guide to using the tool, highlighting its contribution to cybersecurity education and awareness. By leveraging this tool, users can gain insights into the importance of robust password management and adopt best practices to safeguard their digital assets.

1.1 Objective

The primary objective of this project is to develop a Password Cracking Tool that serves as an educational and analytical platform for understanding the security of passwords in the digital landscape. The tool aims to replicate real-world password-cracking scenarios within a safe and controlled environment, allowing users to evaluate the robustness of password-protection methods. By simulating cracking techniques, the project seeks to highlight vulnerabilities associated with weak or predictable passwords, emphasizing the risks they pose to personal and organizational security. Additionally, the project underscores the importance of crafting strong, resilient passwords to counteract such attacks. It also promotes cybersecurity awareness by offering tools to analyse password strength and demonstrating how attackers exploit common patterns and deficiencies in password practices. This project is not intended to aid in unauthorized hacking but to educate users, researchers, and cybersecurity enthusiasts on how password breaches occur and how to prevent them.

Comprehensive Approach:

The key concepts of the project revolve around enhancing password security and promoting cybersecurity awareness by integrating advanced cryptographic principles and practical tools. At its core, the project highlights the role of hashing algorithms as a one-way cryptographic process that secures passwords by converting plain text into fixed-length strings, ensuring that the original password cannot be feasibly retrieved. This foundational concept underscores the importance of safeguarding sensitive information in digital systems. Furthermore, the project delves into the realm of password cracking techniques, focusing on the vulnerabilities of weak passwords. It examines dictionary attacks, which leverage precompiled lists of commonly used passwords to swiftly compromise credentials, and brute force attacks, which involve systematically trying every possible combination to crack passwords. These techniques emphasize the necessity of creating strong, unique, and complex passwords to withstand such attacks.

In addition to highlighting vulnerabilities, the project offers solutions through its password strength analysis module. This feature evaluates passwords based on critical parameters such as length, diversity of characters, and resistance to common attack patterns, providing users with actionable feedback to enhance the robustness of their credentials. By doing so, it not only educates users about secure password creation but also helps them implement stronger defence mechanisms in real-world scenarios. The project also integrates cryptographic concepts by

demonstrating the critical role of algorithms in securing digital systems. It explains how hashing mechanisms operate and illustrates the difficulty of reverse-engineering hashed passwords, thereby reinforcing the importance of robust cryptographic practices to safeguard sensitive data against unauthorized access.

Moreover, the project prioritizes fostering cybersecurity awareness and promoting best practices. It encourages users to adopt proactive measures such as creating long, complex, and unpredictable passwords while avoiding common patterns and reusing credentials. By offering insights into the mindset and methodologies of attackers, the project empowers users to better understand potential threats and develop strategies to mitigate them effectively. Ultimately, this project serves as a comprehensive tool that not only strengthens password security but also educates users about the principles and practices required to build a secure digital environment.

1.2 key features

The Password Cracking Tool integrates a range of features designed to enhance the understanding of password vulnerabilities, demonstrate real-world attack methodologies, and educate users on improving password security. Each feature is carefully tailored to provide both functional and educational value, as outlined below.

1.2.1 Multi-Algorithm Hash Support

This feature ensures the tool is versatile in simulating password cracking scenarios by supporting multiple hashing algorithms used in modern systems:

- 1. MD5:**

A commonly used hashing algorithm known for its speed but also its vulnerabilities to collision attacks. For example, testing the hash of the password "123456" against MD5 demonstrates how quickly weak passwords can be cracked.

- 2. SHA-1:**

While more secure than MD5, SHA-1 has known vulnerabilities. Demonstrating a dictionary attack against SHA-1 shows how slightly stronger hashes can still be compromised.

- 3. SHA-256 & SHA-3:**

Modern, robust algorithms widely used in security applications. Brute-forcing even a short password hashed with SHA-256 illustrates the computational effort required for secure hashes.

4. **BLAKES2:**

A high-performance hash function used in security-sensitive applications, showcasing a balance between speed and cryptographic strength.

1.2.2 Simulation of Attack Methods:

The tool provides a realistic simulation of the most common password cracking methods, helping users understand how attackers exploit password vulnerabilities.:

1. **Dictionary Attacks:**

Dictionary attacks are a prevalent method used in password cracking that leverage precompiled wordlists containing commonly used passwords or predictable combinations. These wordlists often include entries such as “password123,” “admin2024,” or variations of names, dates, and other easily guessable patterns. The attack systematically compares each entry in the wordlist against the target hash to determine if there’s a match. This approach is particularly effective against weak passwords that lack complexity or are based on common terms, phrases, or patterns.

The efficiency of dictionary attacks highlights the inherent risks of using predictable passwords. For instance, passwords derived from simple numerical sequences, birthdays, or common words are highly vulnerable to such attacks, as they are almost always included in standard wordlists. By exploiting this predictability, attackers can often bypass security measures with minimal computational effort.

The use of dictionary attacks underscores the importance of creating strong, unique passwords that include a combination of uppercase and lowercase letters, numbers, and special characters. Users are encouraged to avoid common patterns and to rely on password generators or best practices for crafting secure credentials. This method not only provides an educational perspective on the vulnerabilities of weak passwords but also emphasizes the critical need for robust password hygiene in safeguarding digital systems.

2. **Brute Force Attacks:**

Brute force attacks are a comprehensive and systematic approach to password cracking that attempts to generate and test all possible combinations of characters until the correct one is found. This method exhaustively tries every combination, ranging from simple sequences like “aaaa” to more complex ones such as “zzz9”, encompassing

all possible permutations of letters, numbers, and special characters. The attack continues until it successfully matches the target password, which can be a time-consuming process depending on the complexity of the password and the computational power available to the attacker. One of the primary advantages of brute force attacks is that they do not rely on the target password being weak or predictable. Instead, they methodically cover every possibility, which makes them ideal for testing the impact of password complexity. By evaluating how factors like password length, character variety, and the inclusion of special characters influence security, brute force attacks offer a valuable insight into the effectiveness of different password construction strategies. The longer and more varied the password, the more combinations the attacker must try, significantly increasing the time and computational resources needed to crack it.

This method serves as a critical reminder of the importance of strong passwords in protecting digital assets. For instance, a password that is too short or contains only letters and numbers can be cracked relatively quickly through brute force. However, a longer password that incorporates a mix of uppercase and lowercase letters, numbers, and special symbols exponentially increases the number of combinations an attacker must process. While brute force attacks are effective against all passwords, regardless of their complexity, they highlight the importance of creating passwords that are both long and diverse in character composition. This not only makes brute force attacks more difficult but also acts as a deterrent against attackers, as the time required to crack a robust password can increase exponentially. In essence, brute force attacks are a powerful tool for understanding the fundamental principles of password security and the role that complexity plays in defending against unauthorized access.

3. Password Strength Analyzer:

The Password Strength Analyzer is a critical feature designed to evaluate the robustness of user-provided passwords based on specific criteria. This tool not only assesses the security of passwords but also educates users on creating credentials that are resistant to common attacks. By analysing passwords for factors such as length, the inclusion of special characters, and the avoidance of predictable patterns, the analyser provides a comprehensive evaluation of password strength. It offers users real-time visual feedback through color-coded strength indicators—red for weak passwords,

yellow for moderate ones, and green for strong passwords—making it easier to understand the results at a glance. Additionally, the feature provides tailored recommendations to improve the security of weak passwords, guiding users toward better password practices.

For instance, if a user inputs a password like "12345," the analyser identifies it as weak and provides feedback such as "Weak: Avoid common numerical patterns." It then offers specific recommendations, such as "Add uppercase letters, symbols, and increase the length." This feedback not only highlights the flaws in the password but also gives actionable steps to enhance its security. On the other hand, if the user inputs a stronger password like "A1b@c3d!", the analyser evaluates it as strong, providing feedback such as "Strong: Good mix of characters and sufficient length." This positive reinforcement validates the user's efforts and encourages the continued use of secure password practices.

By combining detailed analysis, intuitive visual feedback, and practical recommendations, the Password Strength Analyzer serves as both a security tool and an educational resource. It empowers users to understand the principles of secure password creation, fostering better cybersecurity habits and enhancing overall digital safety.

4. Custom Wordlist Generator

The Custom Wordlist Generator is a versatile feature that enables users to create personalized wordlists tailored to specific scenarios, making it an invaluable tool for password testing and penetration testing tasks. By allowing users to define input patterns such as names, birthdays, and commonly used suffixes like "123," the generator produces realistic password combinations that simulate those often chosen by individuals. This capability is particularly useful for creating targeted wordlists that align with specific environments or use cases, enhancing the effectiveness of dictionary attacks in ethical hacking and cybersecurity research. The generator supports the inclusion of special characters and variations in letter case, enabling the creation of more complex and realistic wordlists. For instance, users can specify parameters such as a name and a year—for example, "John" and "1990."

The tool then generates a comprehensive list of combinations, including variations like "john1990," "John1990!", and "john@1990." These combinations mimic real-

world password structures, making the tool practical for assessing the security of systems with predictable password patterns. Furthermore, the feature allows users to save the generated wordlists for future use, streamlining workflows and enhancing efficiency in repeated testing scenarios.

By providing a customizable, user-friendly interface and robust functionality, the Custom Wordlist Generator empowers users to craft wordlists that cater to specific testing needs. Its ability to simulate real-world password usage makes it a critical asset for penetration testers and security professionals seeking to identify vulnerabilities and strengthen cybersecurity defences.

5. GUI-Based Design:

The tool features a well-designed Graphical User Interface (GUI) built using Tkinter, providing an intuitive and user-friendly platform accessible to individuals with varying levels of technical expertise. The interface is thoughtfully structured with a clear layout that separates key functionalities into distinct tabs, including hash selection, attack method configuration, and results display. This organized approach ensures that users can navigate the tool effortlessly and focus on specific tasks without confusion. To enhance the user experience, the GUI includes real-time indicators such as progress bars that visually represent the status of ongoing attacks. These indicators provide immediate feedback, allowing users to monitor operations in real time and understand the tool's progress during complex processes like dictionary or brute force attacks. Additionally, robust input validation mechanisms are implemented to ensure that users provide valid data—such as correct hash formats or properly formatted wordlists—before initiating any operations. This feature minimizes errors and streamlines workflows, making the tool both reliable and efficient. Upon launching the tool, users are greeted with a welcoming screen that offers options for selecting hash algorithms and inputting passwords or wordlists. The interface is designed for seamless navigation, guiding users step-by-step through the process with clarity and precision. By combining functionality with an intuitive design, the GUI not only enhances accessibility but also ensures that users can leverage the tool's capabilities effectively, regardless of their technical background.

6. Educational Focus:

Designed with a strong emphasis on learning and awareness, the tool serves as

both a functional application and an educational resource, fostering a deeper understanding of cybersecurity principles. It includes interactive features that provide users with comprehensive explanations for each attack method and its implications. For example, users can explore how dictionary and brute force attacks work, gaining insights into the vulnerabilities these methods exploit. By presenting this information in a clear and accessible manner, the tool aims to demystify complex cybersecurity concepts and promote informed decision-making.

Additionally, the tool integrates tutorials on creating strong passwords and understanding the basics of cryptographic principles, empowering users to enhance their digital security practices. Features like tooltips offer contextual guidance, ensuring that users can learn as they navigate through the interface. For instance, a tooltip next to the “Dictionary Attack” option might explain: “Dictionary attacks exploit commonly used passwords. Use unique, complex passwords to protect your accounts.” This real-time educational feedback not only increases user engagement but also reinforces best practices in password management.

By combining interactive features with practical guidance, the tool transforms into a dynamic learning platform. It not only performs security operations but also educates users on safeguarding their digital assets, making it an invaluable resource for anyone looking to strengthen their cybersecurity knowledge.

CHAPTER 2

GUI DESIGN AND STRUCTURE

The interface is designed to be intuitive, ensuring that users of all experience levels can easily navigate to the desired module with a single click. The responsive layout and consistent styling provide a seamless user experience, setting the tone for the tool's purpose of simplifying password-related tasks in an efficient and visually appealing manner. The Main Page acts as the gateway to the tool's comprehensive features, allowing users to perform password attacks, generate secure passwords, create custom wordlists, or analyze password strength. The Main Page of the Password Cracking Tool serves as the central navigation hub, offering users quick access to all its functionalities through five distinct buttons: Dictionary Attack, Brute Force Attack, Password Generator, Wordlist Generator, and Password Analyzer. Each button is prominently displayed in a vertical stack on a clean, dark background for visual clarity and ease of interaction.

2.1 Main Page:

The main page serves as the central navigation hub for the application, providing users with an intuitive and visually appealing interface to access its core functionalities. At the top centre of the page, a prominently displayed title label, "Password Cracking Tool", immediately establishes the purpose of the tool. This title is styled in a large, bold font to capture the user's attention and reinforce the application's branding. Positioned beneath the title are five interactive buttons—Dictionary Attack, Brute Force Attack, Password Generator, Wordlist Generator, and Password Analyzer—each representing a key feature of the application. These buttons are arranged in a clean, vertical stack at the centre of the screen, ensuring ease of access and intuitive navigation.

To enhance usability, the main page design incorporates a thoughtful colour scheme, featuring a dark background that creates a striking contrast with the bright, vibrantly coloured buttons. This not only improves visibility but also adds a modern, professional aesthetic to the interface. Each button is styled with a consistent font, size, and colour palette, creating a cohesive and polished appearance that aligns with the overall theme of the application. Furthermore, the page is designed to be highly responsive, ensuring optimal performance across different screen sizes and resolutions.

The simplicity and clarity of the layout minimize cognitive load, allowing users to quickly identify and select their desired functionality. The dark background, in combination with the clear fonts and button designs, reduces eye strain and enhances the overall user experience. By seamlessly blending functionality with modern design principles, the main page sets the tone for the rest of the application, offering a user-friendly and visually engaging entry point to its diverse features. This thoughtful design ensures that users of all experience levels can easily navigate the application while appreciating its polished and professional interface.

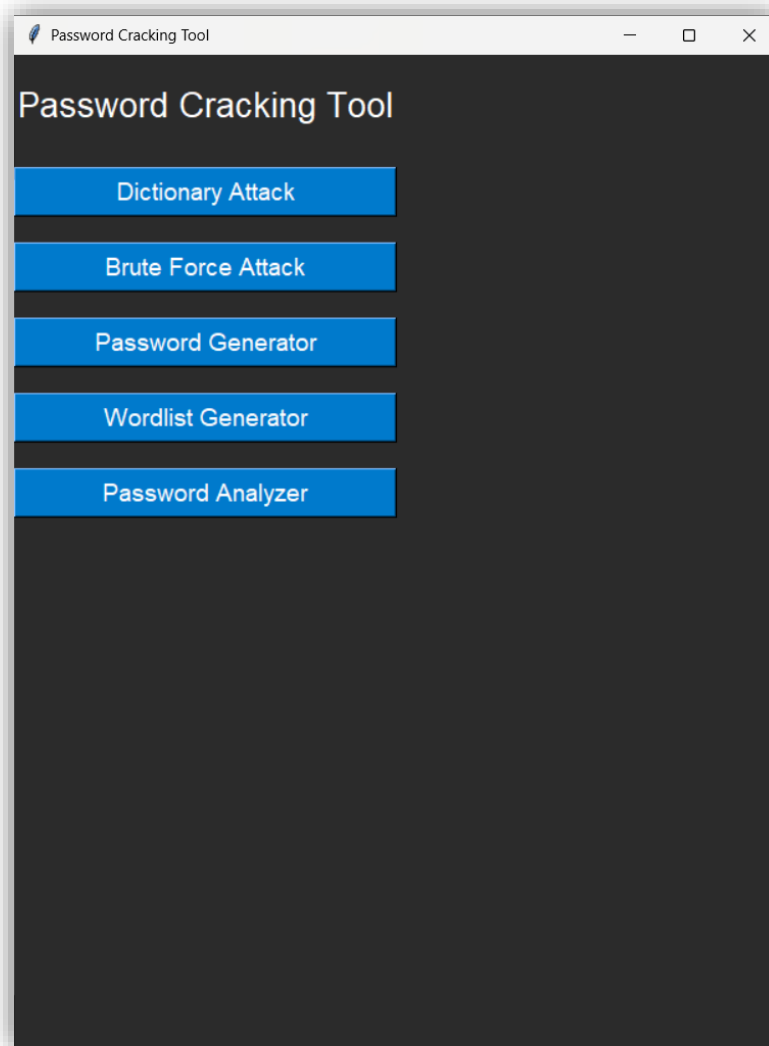


Figure1: Screenshot of Main window

2.2 Button Functionality

The buttons in the GUI are designed to trigger specific actions and provide the user with intuitive navigation through the tool's features. Each button is carefully placed to be easily accessible, and their functions are clearly labeled for better usability.

1. Dictionary Attack:

The Dictionary Attack module is a critical feature of the application, designed to facilitate password cracking using a user-provided dictionary file. This module is purpose-built to demonstrate how weak passwords can be uncovered by systematically matching hashes against entries in a dictionary file. The interface is thoughtfully organized with various widgets to ensure ease of use and functionality. At the top of the module, users will find clearly labelled input fields. The first field, titled "Hash", allows users to input the target hash that they wish to crack. Adjacent to it, another entry field labelled "Dictionary File" includes a convenient "Browse" button, enabling users to upload a dictionary file directly from their system. This streamlined design simplifies the process of preparing the module for operation.

Beneath these input fields, a dropdown menu is provided, allowing users to select the desired hash function for the attack, such as MD5, SHA-1, or SHA-256. This ensures compatibility with various hashing algorithms and broadens the module's applicability. Below the dropdown menu, three essential buttons are displayed: "Start Dictionary Attack" initiates the attack, "Stop" halts the process if necessary, and "Back to Main Menu" enables users to return to the application's main page effortlessly. These buttons are clearly labelled and strategically positioned for quick and intuitive access.

To enhance user experience further, the module includes a real-time log display area, which is situated below the input fields and buttons. This text area provides continuous updates on the progress of the dictionary attack, including details such as attempted passwords and their corresponding hashes. By offering real-time feedback, the tool keeps users informed and engaged throughout the process.

The overall layout is designed to resemble a form-like structure, ensuring that inputs and controls are logically grouped and easy to navigate. The real-time log display is seamlessly integrated beneath the form, maintaining a clean and organized interface. This thoughtful arrangement not only enhances usability but also provides a professional and polished appearance, making the module accessible to users of all experience levels. By combining functionality with an intuitive design, the Dictionary Attack module serves as an effective tool for demonstrating password security vulnerabilities and the importance of robust password practices.

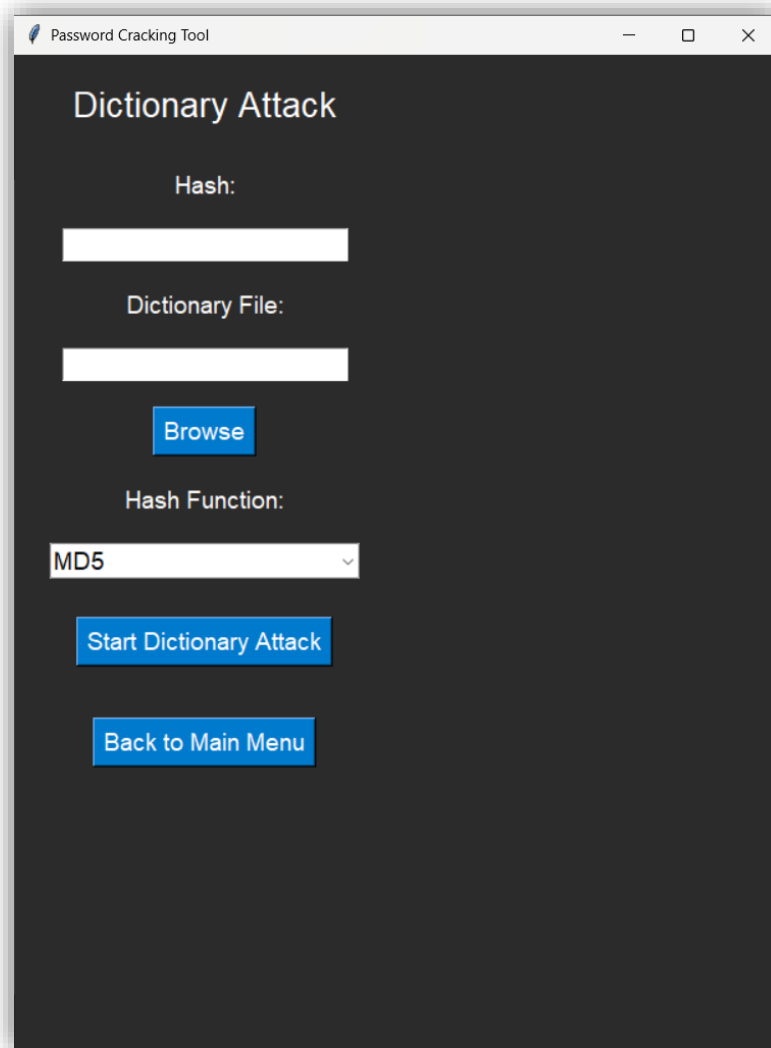


Figure 2: Dictionary Attack

2. **Brute Force Attack:**

The Brute Force Attack module is a powerful feature of the application, designed to systematically generate and test all possible password combinations to crack a given hash. This module is particularly useful for demonstrating the resilience of strong passwords and the sheer computational effort required to crack them through exhaustive methods. The interface has been meticulously crafted to provide a balance of functionality and user-friendliness, ensuring that users can easily operate the tool while gaining valuable insights into brute force attack mechanics.

At the top of the module, users will find the primary input fields for configuring the attack. The first field, labelled "Hash", allows users to input the hash they wish to crack. This field supports various hash formats, making it versatile for different

scenarios. Next to this, the "Max Length" input field enables users to specify the maximum length of the password combinations to be tested. This feature is particularly important as it allows users to control the complexity of the attack, balancing comprehensiveness with computational feasibility. These input fields are clearly labelled to ensure clarity and ease of use, even for users with limited experience in password security tools.

Below the input fields, the module features a dropdown menu that provides a list of supported hash functions, including MD5, SHA-1, and SHA-256. This menu ensures compatibility with widely used cryptographic standards, allowing users to tailor the brute force attack to the specific algorithm used to generate the target hash. This flexibility enhances the module's applicability and reinforces its role as a comprehensive tool for password testing and education.

Directly beneath the dropdown menu are three key action buttons that enable users to control the attack process. The "Start Brute Force Attack" button initiates the systematic testing of all possible password combinations based on the selected parameters. This feature is complemented by the "Stop" button, which gives users the ability to halt the attack at any time, providing control and flexibility during operations. The "Back to Main Menu" button ensures seamless navigation by allowing users to exit the module and return to the main interface effortlessly. These buttons are clearly labelled and thoughtfully arranged for ease of access, ensuring an intuitive user experience.

To provide users with real-time feedback, the module includes a progress bar, positioned at the bottom of the interface. This visual element dynamically updates to reflect the ongoing status of the brute force attack, offering a clear indication of progress and expected completion time. The progress bar not only enhances the user experience by adding a layer of interactivity but also serves as a practical tool for monitoring the attack's efficiency.

The overall layout of the module is designed with simplicity and functionality in mind. Input fields are placed prominently at the top of the interface to prioritize user configuration, followed by the action buttons for streamlined operation. The progress bar is strategically positioned at the bottom to provide continuous feedback without cluttering the main workspace. This hierarchical arrangement ensures that the interface is easy to navigate while maintaining a professional and polished appearance.

By systematically testing all possible combinations, the Brute Force Attack module demonstrates the computational intensity of this method and emphasizes the importance of creating strong, complex passwords. It serves as both an educational and practical tool, allowing users to understand the mechanics of brute force attacks and the critical role of password length and complexity in resisting such attempts. The module's robust design, combined with its user-friendly interface, ensures that it is a valuable addition to the application, providing insights into the challenges of password security in the digital age.

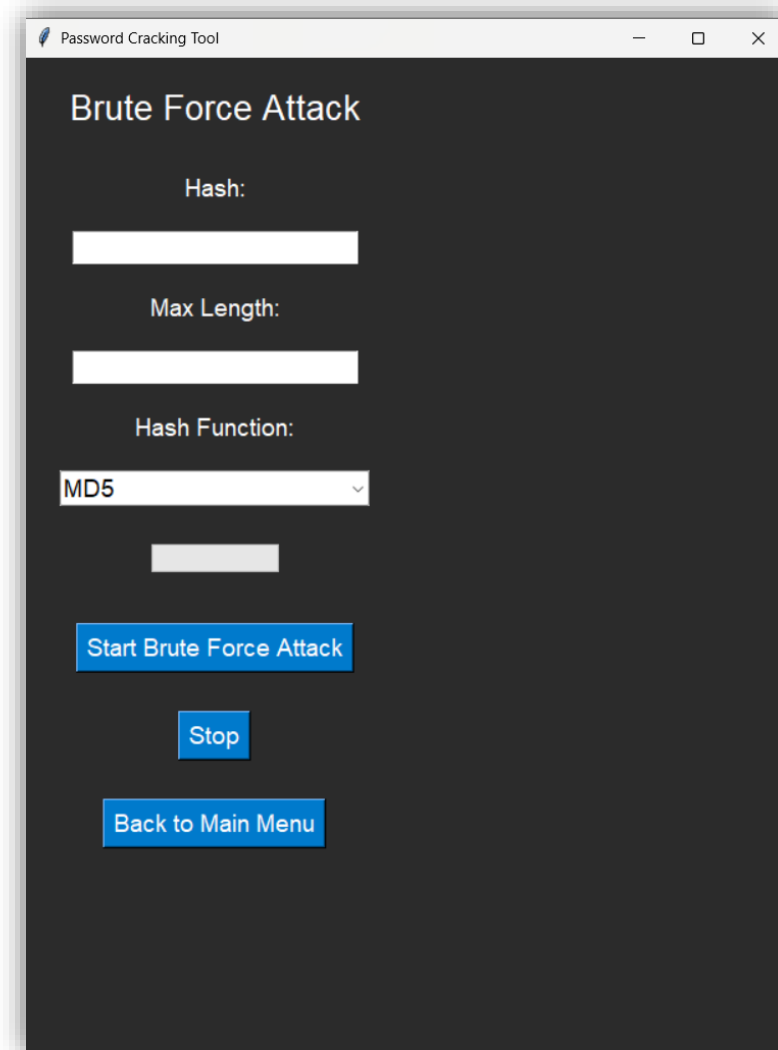


Figure3: Brute Force Attack

3. Password Generator:

The Password Generator module is a versatile and user-friendly feature designed to create strong, secure passwords based on user-defined criteria. It is an essential component of the application, providing users with the ability to generate

robust passwords that meet specific security requirements. This module not only simplifies the process of creating secure credentials but also educates users about the importance of incorporating diverse character types and adequate length into their passwords to resist potential attacks. At the core of the interface are input widgets that allow users to customize the password generation process. The "Length" slider is prominently positioned, enabling users to select the desired length of the password with precision. This slider is intuitive and allows for a broad range of lengths to cater to various security needs, from short but complex passwords to lengthy and highly secure ones. Below the slider, a series of checkboxes provide users with additional customization options. These checkboxes include "Include Uppercase Letters," "Include Lowercase Letters," "Include Numbers," and "Include Special Characters." By selecting these options, users can tailor the complexity of the generated password to meet specific requirements, such as those imposed by online services or organizational policies.

Beneath the customization options, the module includes three interactive buttons that enhance usability and functionality. The "Generate Password" button initiates the password creation process based on the selected criteria, instantly providing users with a secure, randomly generated password. Once the password is generated, users can click the "Copy Password" button to copy it directly to their clipboard, making it easy to paste into applications, websites, or password management tools. For smooth navigation, the "Back to Main Menu" button is also included, allowing users to return to the main interface without hassle.

The generated password is displayed in a text area located at the bottom of the module. This dedicated space ensures that the password is prominently visible and easy to review before use. The text area is styled to accommodate various password lengths while maintaining clarity and readability. This design element not only enhances user experience but also ensures that users can confirm the password's composition and strength before copying or implementing it.

The layout of the Password Generator module is clean, organized, and intuitive. The slider and checkboxes are vertically arranged to create a logical flow, guiding users step-by-step through the customization process. The text area for displaying the generated password is strategically placed at the bottom of the interface, ensuring that users can view the final result without unnecessary scrolling or searching. This vertical

arrangement maintains a user-friendly interface while promoting efficiency and accessibility.

Overall, the Password Generator module combines simplicity with functionality, making it an invaluable tool for creating secure passwords. By providing customizable options and an intuitive interface, it empowers users to create strong passwords tailored to their specific needs. Additionally, it reinforces best practices in password security by highlighting the importance of incorporating uppercase and lowercase letters, numbers, special characters, and adequate length into password creation. This module not only simplifies the process of generating secure passwords but also helps users understand the critical role of password complexity in maintaining digital security. Its robust design and educational value make it a key feature of the application, catering to both novice and experienced users.

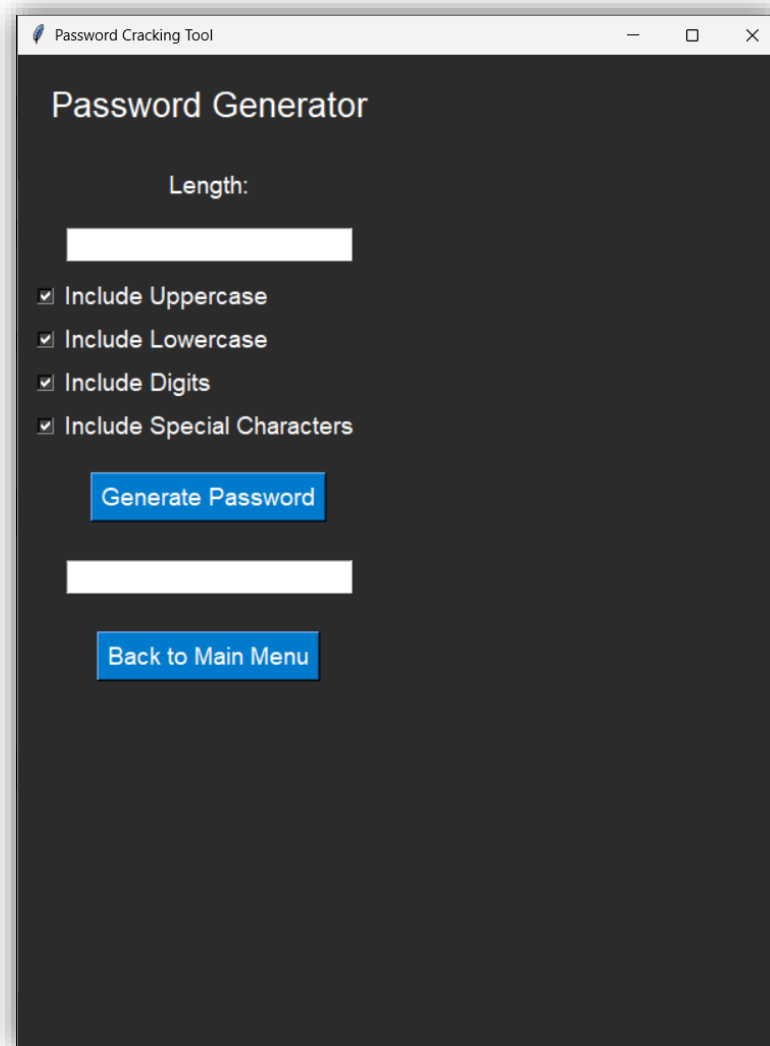
The image shows a screenshot of a software application window titled "Password Cracking Tool". Inside the window, there is a section titled "Password Generator". Below this title, there is a label "Length:" followed by a white rectangular input field. Underneath the input field, there are four checkboxes, all of which are checked: "Include Uppercase", "Include Lowercase", "Include Digits", and "Include Special Characters". Below these checkboxes is a blue button with the text "Generate Password". Underneath the button is another white rectangular input field. At the bottom of the section is another blue button with the text "Back to Main Menu". The entire interface is set against a dark gray background.

Figure 4: Password Generator

4. Wordlist Generator:

The Wordlist Generator module is a versatile and practical feature designed to create custom wordlists based on user-defined patterns. This module is particularly valuable for security enthusiasts, ethical hackers, and researchers who need tailored wordlists for tasks such as password cracking, penetration testing, or vulnerability analysis. By allowing users to define specific input patterns, the module ensures flexibility and efficiency in generating wordlists that align with the user's unique requirements.

At the top of the interface, the module features a prominently placed "Input Pattern" field. This field enables users to input custom patterns or rules that dictate how the wordlist should be generated. For example, users can specify combinations of letters, numbers, or symbols, or include placeholders for dynamic variables. This level of customization ensures that the wordlist can be tailored to specific contexts, such as testing the security of systems with predictable password structures. The input pattern field is designed to be intuitive, with clear labelling and formatting guidance to assist users in crafting effective patterns.

Beneath the input field, the module includes three interactive buttons that streamline the wordlist generation process. The "Generate Wordlist" button initiates the creation of the wordlist based on the defined input pattern, producing a comprehensive list of possible combinations. Once the wordlist is generated, users can click the "Save Wordlist" button to store the generated list as a file on their local device. This feature is especially useful for users who need to reuse or analyse the wordlist in other applications or tools. Additionally, the "Back to Main Menu" button allows users to exit the module and return to the main interface effortlessly, ensuring smooth navigation and usability.

To enhance interactivity and provide real-time feedback, the module includes a text area at the bottom of the interface. This area displays a preview of the generated wordlist, allowing users to review its content before saving or using it. The preview feature ensures that users can verify the accuracy and relevance of the wordlist, making adjustments to the input pattern if necessary. The text area is styled to handle large datasets while maintaining readability, ensuring that even extensive wordlists can be easily reviewed. The layout of the module is clean and logically organized for maximum usability. The input field is positioned at the top of the interface, prioritizing user input and customization. The buttons are aligned directly below the input field,

ensuring quick access to the module's primary functions. Finally, the wordlist preview is displayed at the bottom of the interface, providing a clear and accessible view of the generated output. This hierarchical arrangement ensures a seamless and efficient workflow, guiding users step-by-step through the wordlist generation process.

The Wordlist Generator module is designed to balance simplicity and functionality, making it an essential tool for tasks that require tailored wordlists. By offering customizable patterns, real-time previews, and efficient saving options, it empowers users to create wordlists that are both precise and comprehensive. This module not only enhances the application's utility but also serves as an educational tool, demonstrating the importance of targeted wordlists in password testing and security assessments. Its robust features, combined with an intuitive interface, make it a valuable resource for both beginners and professionals in the field of cybersecurity.

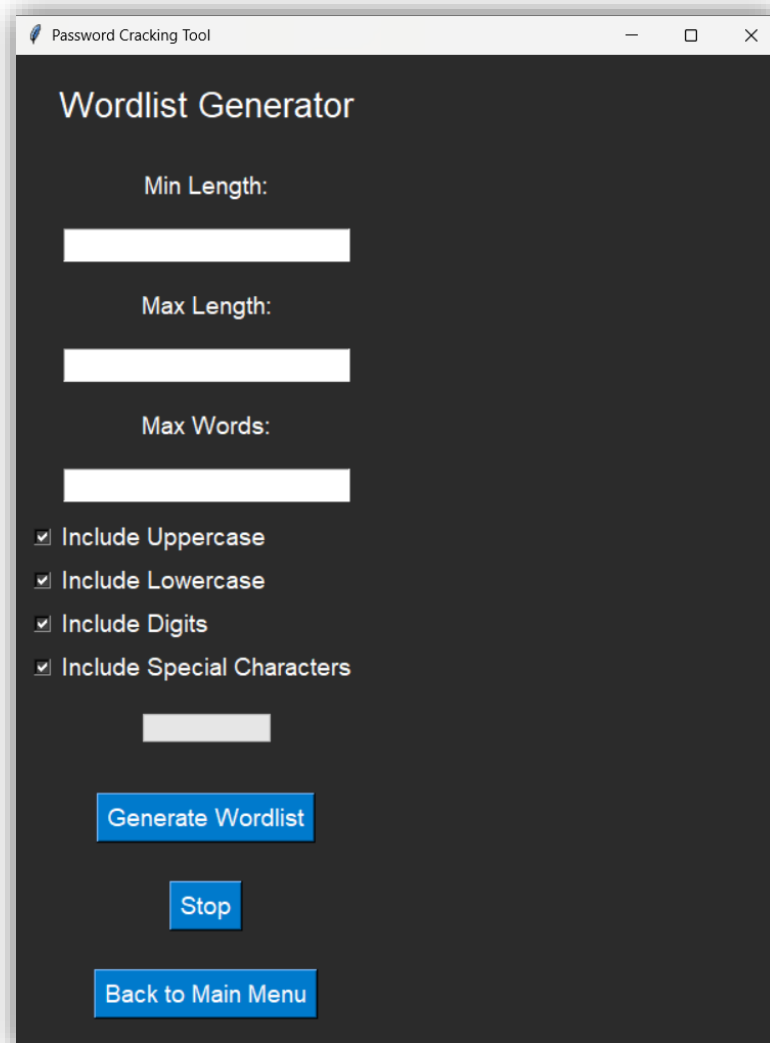
The image shows a software window titled "Password Cracking Tool" with a dark theme. Inside the window is a section titled "Wordlist Generator". It contains three input fields for "Min Length:", "Max Length:", and "Max Words:". Below these are four checked checkboxes: "Include Uppercase", "Include Lowercase", "Include Digits", and "Include Special Characters". There is a small progress bar below the checkboxes. At the bottom of the section are three buttons: "Generate Wordlist" (highlighted in blue), "Stop", and "Back to Main Menu" (highlighted in blue).

Figure5: Wordlist Generator

5. Password Analyzer:

The Password Analyzer module is an integral part of the application, meticulously designed to evaluate the strength of user-provided passwords and offer actionable insights for enhancing their security. This module aims to educate users about the principles of secure password creation by analysing various aspects of a password's composition, such as length, character diversity, and resistance to common attack vectors. By providing a clear evaluation and tailored recommendations, the module empowers users to make informed decisions about password security.

At the heart of the module is the password input field, where users can enter the password, they wish to analyse. This input field, labelled clearly as "Password", is designed to be intuitive and accessible, ensuring users can quickly and easily input their passwords for analysis. The field supports a wide range of password types, from simple alphanumeric combinations to complex strings with special characters, making it versatile for different security scenarios.

Beneath the input field are two primary buttons that control the module's functionality. The "Analyse" button initiates the evaluation process, applying a set of predefined criteria to assess the password's strength. These criteria include factors such as length, the inclusion of uppercase and lowercase letters, the presence of numbers and special characters, and resistance to dictionary or brute force attacks. Once the analysis is complete, the results are displayed in real-time, providing users with immediate feedback. The second button, "Back to Main Menu," offers seamless navigation, allowing users to exit the module and return to the main interface without losing their progress. This intuitive button placement ensures a smooth and efficient user experience.

To present the analysis results, the module features a text box positioned prominently below the input field. This text box is the primary output area, displaying the password's strength score on a predefined scale, such as weak, moderate, strong, or very strong. In addition to the score, the text box provides detailed recommendations for improving password security. For example, if the password lacks special characters or is too short, the module suggests adding more complexity or increasing its length. By offering specific and actionable feedback, the module not only evaluates the password but also guides users in creating more secure alternatives. The layout of the Password Analyzer module is thoughtfully designed to ensure clarity and usability. The password input field is positioned at the top of the interface, prioritizing user input.

Directly beneath it, the buttons are strategically placed for easy access, streamlining the analysis process. The text box, which displays the results, occupies the bottom section of the interface, ensuring that the output is prominently visible without overcrowding the workspace. This logical arrangement creates a cohesive flow, guiding users step-by-step through the evaluation process.

Beyond its functional design, the Password Analyzer module serves as a vital educational tool, highlighting the importance of strong, unique passwords in protecting sensitive information. By analysing passwords against common attack methods and providing tailored advice, it reinforces best practices for password security. The module's interactive nature and clear feedback mechanisms make it accessible to users of all skill levels, from beginners seeking to understand password basics to advanced users aiming to fine-tune their security measures. With its robust features and intuitive interface, the Password Analyzer module is a cornerstone of the application, bridging the gap between password evaluation and education. It empowers users to proactively improve their password practices, fostering a deeper understanding of cybersecurity principles and enhancing overall digital security.

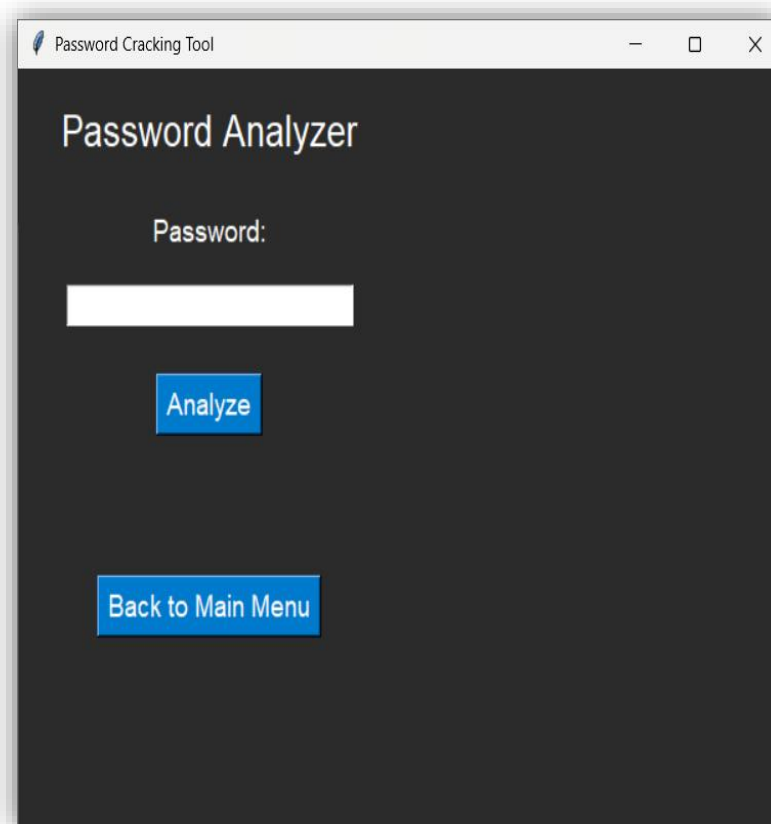


Figure 6: Password Analyzer

CHAPTER 3

WORKFLOW AND SYSTEM OVERVIEW

The Password Cracking Tool is a robust application designed to simulate and analyze password security by employing various cryptographic techniques and algorithms. It enables users to generate secure passwords, analyze existing passwords for strength, create custom wordlists, and perform dictionary and brute-force attacks on password hashes. This chapter provides a comprehensive overview of the tool's workflow, detailing its usage and methodology for executing its core functionalities.

3.1 Workflow

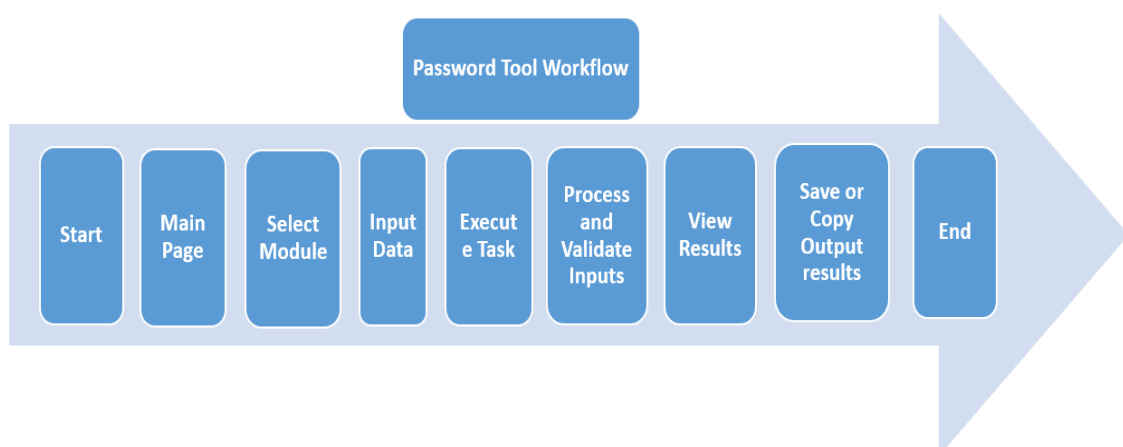


Figure 7: Workflow diagram

1. **Start:** Launch the application (e.g., double-click on `PasswordCracker.exe`).
2. **Main Page :** Access the main page with functional tabs:
 - a. Dictionary Attack
 - b. Brute Force Attack
 - c. Password Generator
 - d. Wordlist Generator
 - e. Password Analyzer
3. **Select Module:** Click on the desired module.
4. **Input Data:** Provide necessary details like hash values, parameters, or wordlist paths.
5. **Execute Task:** Click "Start" to run the operation.
6. **Process and Validate Inputs:** Application processes the input and flags errors if any.
7. **View Results:** Results are displayed (e.g., cracked passwords, password strength).
8. **Save or Copy Output:** Save results for future use or copy for analysis.
9. **End**

3.2 System Overview:

The Password Cracking Tool operates on a modular design that integrates its five core functionalities. The design ensures flexibility, scalability, and ease of use while providing a secure environment for testing password vulnerabilities. The tool is designed to enhance password security by providing functionalities such as password cracking, generating secure passwords, creating wordlists, and analyzing password strength.

3.2.1 Key Components:

1. User Interface (UI):

The interface is designed to be intuitive, ensuring that users of all experience levels can easily navigate to the desired module with a single click. The responsive layout and consistent styling provide a seamless user experience, setting the tone for the tool's purpose of simplifying password-related tasks in an efficient and visually appealing manner.

- Five functional tabs:
 - a. Dictionary Attack
 - b. Brute Force Attack
 - c. Password Generator
 - d. Wordlist Generator
 - e. Password Analyzer

2. Processing Engine:

The core system logic serves as the backbone of the tool, executing critical tasks such as cracking hashes, generating secure passwords, and analyzing password strength. It incorporates advanced algorithms for brute force and dictionary attacks, ensuring effective and reliable performance across various password security scenarios.

3. Input Validation Module:

The system includes a robust input validation mechanism that ensures user-provided data, such as hash values, file paths, and parameters, is correctly formatted. This feature minimizes errors during processing, enhancing the tool's reliability and usability.

4. **Output Display and Reporting:**

The tool presents results in a clear and organized format, making it easy for users to interpret findings. It also provides options to save or copy outputs, such as cracked passwords or wordlist summaries, for further analysis or documentation.

5. **System Architecture:**

The tool's modular design ensures scalability and simplifies maintenance, allowing for seamless integration of new features and updates. Interaction between the front-end user interface and back-end logic is managed through event-driven programming, enabling efficient and responsive performance.

6. **Security Features:**

The tool prioritizes security by ensuring that sensitive data, such as passwords, is not stored unnecessarily. It also incorporates safeguards to prevent misuse of its functionalities, maintaining ethical and secure operation standards.

3.2.2 **System Workflow Overview**

1. User launches the application and selects a tab.
2. Inputs are validated before processing.
3. The processing engine executes the task using the selected module.
4. Results are displayed or exported for further use.

3.3.3 **System Specifications Requirements:**

To ensure smooth functioning, the tool requires the following system specifications:

- **Operating System:** Windows, macOS, or Linux.
- **RAM:** Minimum 4GB.
- **Processor:** intel i5 or equivalent.
- **Python Libraries:** skinter, harshly, random intercools.

CHAPTER 4

DATA COLLECTION METHODOLOGIES

Data collection forms the backbone of the Password Cracking Tool, as it gathers, processes, and analyses password-related information to provide critical insights into system security. The tool uses systematic approaches for collecting data, which include retrieving password hashes, generating wordlists, and analysing password strength. This chapter provides a detailed explanation of the methodologies implemented for efficient and secure data collection in the tool.

4.1 Password Hash Collection

Password hashes are the primary input for cracking attempts in the Dictionary Attack and Brute Force Attack modules. The following steps outline how password hashes are collected and utilized:

1. User Input:

- The user manually provides a password hash in the designated text field.
- Supported hash formats include MD5, SHA-1, SHA-256, SHA-3, and BLAKE2.

2. Hash Validation:

- The tool validates the format of the entered hash to ensure compatibility with the cracking algorithms.
- Invalid or unsupported hashes trigger an error message, prompting the user for correction.

3. Secure Handling:

- Password hashes are temporarily stored in memory for processing and are not written to disk to ensure privacy and security.

4.2 Wordlist Collection

Wordlists are a critical resource for the Dictionary Attack module. The tool enables users to either load pre-existing wordlists or generate custom ones:

1. Loading Pre-Existing Wordlists:

- Users can upload wordlist files in standard text format (.txt).
- The tool verifies the integrity of the file and checks for duplicate entries to

enhance cracking efficiency.

2. Custom Wordlist Generation:

- The Wordlist Generator module allows users to create tailored wordlists by specifying parameters like character sets, word length, and patterns.
- The generated wordlists are stored in a structured format and displayed to the user for review.

4.3 Password Strength Data Collection

The Password Analyzer module evaluates the strength of user-provided passwords by analysing their characteristics:

1. Input Requirements:

- The user provides a password for analysis in the designated field.
- The tool does not save or store the entered password, ensuring user privacy.

2. Feature Extraction:

- The tool evaluates various attributes of the password, such as length, use of uppercase/lowercase characters, inclusion of special characters, and entropy.
- These features are collected and processed to calculate the password's overall strength.

3. Strength Metrics:

- The tool provides a detailed breakdown of the password's strengths and weaknesses.
- Suggestions for improvement are provided if the password is deemed weak.

4.4 Brute Force Data Collection

The Brute Force Attack module systematically tests all possible combinations of characters to crack a password. This process involves the following steps:

1. Parameter Configuration:

- The user specifies parameters like character sets (e.g., alphabets, numbers, special characters) and maximum password length.
- These parameters are used to generate all possible combinations within the defined scope.

2. Combination Generation:

- The tool dynamically generates password combinations using Python's intercools library.
- The generated combinations are processed in real-time, ensuring memory efficiency.

3. Result Storage:

- Once the correct password is identified, it is displayed to the user along with details like time taken and number of combinations tried.

4.5 Data Validation and Security

The tool incorporates robust validation and security mechanisms to ensure the reliability of collected data:

1. Input Validation:

- Ensures all inputs, such as hashes, passwords, and wordlist files, meet the required format and constraints.
- Prevents errors and enhances the accuracy of results.

2. Temporary Storage:

- Data like generated combinations and collected wordlists are stored temporarily during processing and deleted upon task completion.

3. Encryption:

- All sensitive data processed by the tool is encrypted to protect it from unauthorized access.

CHAPTER 5

CODE ARCHITECTURE AND IMPLEMENTATION

The Password Cracking Tool is built with a modular and scalable code architecture to support its various functionalities, including password cracking, strength analysis, and wordlist generation. This chapter details the overall structure of the codebase, class architecture, data handling methodologies, and error-handling mechanisms implemented in the tool.

5.1 Class-Based Architecture

The tool's codebase follows a class-based architecture, dividing the application into specific modules. Each class is responsible for a distinct functionality, promoting modularity, maintainability, and scalability.

1. Main Application Class:

The Main Application Class acts as the central hub and entry point of the tool, seamlessly bridging the graphical user interface (GUI) with the backend functionalities. It is responsible for managing the overall flow of the application, ensuring smooth interactions between various modules and maintaining a cohesive user experience. This class plays a critical role in initializing all necessary modules, coordinating their interactions, and ensuring they function harmoniously to deliver the desired results.

A key feature of the Main Application Class is its ability to load the Tkinter-based user interface, providing users with an intuitive platform to access the tool's functionalities. Additionally, it efficiently handles user events, such as button clicks or form submissions, by delegating specific tasks to the appropriate backend modules. This design ensures that the application remains modular, scalable, and easy to maintain, allowing for future enhancements and updates without disrupting the overall structure.

2. Hash Cracking Class:

The Hash Cracking Class is a core component of the tool, implementing the critical logic required for executing Dictionary Attacks and Brute Force Attacks. Its primary responsibility is to manage and apply algorithms for cracking various cryptographic hash functions, including MD5, SHA-1, SHA-256, SHA-3, and BLAKE2, ensuring comprehensive support for different hash types. This class serves as the computational engine behind the tool's password-cracking capabilities, enabling efficient and accurate processing.

Key features of the Hash Cracking Class include its use of both iterative and recursive methods for generating and testing password combinations. For dictionary attacks, it reads precompiled wordlists and systematically matches each entry against the target hash. For brute force attacks, it dynamically generates all possible combinations of characters based on user-defined criteria, testing each combination until a match is found. Additionally, the class tracks performance metrics, such as the time taken and the number of attempts made during the cracking process, providing valuable insights into the efficiency and effectiveness of the operations.

By combining robust algorithms with detailed tracking and flexible methods, the Hash Cracking Class forms the backbone of the tool's functionality, ensuring accurate and efficient password recovery for a wide range of use cases.

3. **Password Generator Class:**

The Password Generator Class is designed to create strong and secure passwords tailored to user-defined criteria, playing a crucial role in promoting better password hygiene. Its primary responsibility is to generate passwords that meet specific requirements, such as desired length and inclusion of various character types. By allowing users to customize the character set—such as including uppercase letters, lowercase letters, numbers, and special characters—the class ensures flexibility while maintaining robust security standards.

A standout feature of this class is its ability to ensure randomness and strength in the generated passwords. By avoiding predictable patterns and leveraging diverse character combinations, it produces passwords that are resistant to common cracking methods like dictionary and brute force attacks. This makes the Password Generator Class an essential tool for users seeking to create unique credentials that effectively safeguard their digital assets.

4. **Wordlist Generator Class:**

The Wordlist Generator Class is a versatile component that facilitates the creation of customized wordlists tailored to specific password-cracking scenarios. Its primary responsibility is to generate wordlists based on user-defined parameters, such as the desired character set and the length of the words. This allows users to create targeted lists that mimic potential passwords, improving the efficiency and precision of dictionary attacks.

One of the key features of this class is its ability to accept and process user inputs to produce wordlists that align with specific needs, such as incorporating certain

patterns or character combinations. Additionally, the class supports saving these generated wordlists in a structured format, enabling users to reuse them in future operations. By combining flexibility with practicality, the Wordlist Generator Class is an invaluable tool for conducting focused and effective password-cracking tasks.

5. Password Analyzer Class:

The Password Analyzer Class is a critical feature that assesses the strength and security of user-provided passwords, helping users understand the vulnerabilities and robustness of their credentials. Its primary responsibility is to evaluate passwords by analysing various attributes, including length, entropy, and character diversity. This thorough analysis ensures that users receive a clear understanding of their password's ability to resist common cracking techniques.

A standout feature of this class is its capability to provide actionable feedback and recommendations for enhancing password security. Based on the analysis, it suggests improvements such as increasing the password length, incorporating a greater variety of characters, or avoiding common patterns. By combining detailed evaluation with practical guidance, the Password Analyzer Class serves as an essential tool for fostering stronger and more secure password practices.

6. Utility Class:

The Utility Class serves as a supportive component within the tool, managing a variety of reusable tasks to ensure smooth and efficient operations. Its primary responsibilities include handling file operations, such as reading and writing wordlists, managing logs, and formatting data for seamless integration across different modules. By centralizing these common tasks, the Utility Class simplifies the overall workflow and reduces redundancy in the codebase.

A key feature of this class is its ability to format output data for display within the GUI, ensuring that information is presented in a clear and user-friendly manner. Whether it's processing wordlists for cracking operations or preparing logs for progress updates, the Utility Class plays a pivotal role in maintaining the tool's functionality and user experience. Its adaptability and efficiency make it an indispensable part of the system's architecture.

5.2 Data Handling

The tool handles data efficiently and securely to ensure smooth operations while protecting user inputs.

1. **Input Handling:**

- Password hashes, user-generated wordlists, and parameters are validated for correctness.
- Input errors (e.g., unsupported hash formats) are flagged immediately to guide the user.

2. **Data Storage:**

- Temporarily stores data in memory during operations.
- Ensures sensitive data, such as password hashes, is not written to disk, maintaining security.

3. **Data Processing:**

- Filters, validates, and organizes collected data for efficient cracking and analysis.
- Dynamically generates password combinations for brute force operations.

4. **Data Output:**

- Results (e.g., cracked passwords, strength analysis) are presented clearly in the GUI.
- Users can export results to text files for further review.

5.3 Error Handling

A robust error-handling mechanism ensures the tool runs reliably and provides meaningful feedback to users.

1. **Try-Except Blocks:**

- Catches exceptions during file operations, invalid user input, or system-related errors.
- Prevents the tool from crashing unexpectedly.

2. **Error Logging:**

- Logs errors with timestamps for debugging purposes.
- Errors are stored in a designated log file for developer review.

3. **User-Friendly Feedback:**

- Displays clear error messages, such as "Invalid hash format" or "Wordlist file not found."
- Guides users to correct issues and retry operations.

4. **Graceful Recovery:**

- Provides options to retry failed tasks or reset modules.
- Ensures no data is lost or corrupted during recovery.

5.4 Code Optimization

The tool incorporates several optimization techniques to improve performance and maintainability:

1. Efficient Algorithms:

- Uses optimized libraries like `harshly` for hash verification and `intercools` for combination generation.
- Reduces redundant computations by caching results where applicable.

2. Modular Design:

- Separates concerns into individual classes and functions, making the codebase easier to debug and extend.

3. Scalability:

- Designed to support additional hash types or cracking methods with minimal changes.
- Easily integrates new features, such as additional password analysis metrics or advanced attack modes.

5.5 GUI Integration

The **Tainter-based GUI** acts as the frontend of the tool, seamlessly integrating with backend logic:

1. Tabbed Layout:

- The GUI includes five tabs:
 - Dictionary Attack
 - Brute Force Attack
 - Password Generator
 - Wordlist Generator
 - Password Analyzer
- Each tab is linked to its respective backend class.

2. Dynamic Updates:

- Displays real-time progress for cracking attempts.
- Shows results and logs dynamically without requiring page reloads.

3. Responsive Design:

- Adapts to different screen resolutions.
- Ensures proper alignment of text boxes, buttons, and scrollbars.

CHAPTER 6

RESULTS

6.1 Result of dictionary attack

The Dictionary Attack successfully recovered the password "cyber" by comparing entries from the provided wordlist against the target hash using the selected hash function (e.g., MD5). Once a match was found, the result was displayed in a popup for confirmation. This demonstrates the tool's efficiency and accuracy in recovering hashed passwords when a suitable wordlist is available, making it valuable for security testing and investigations.

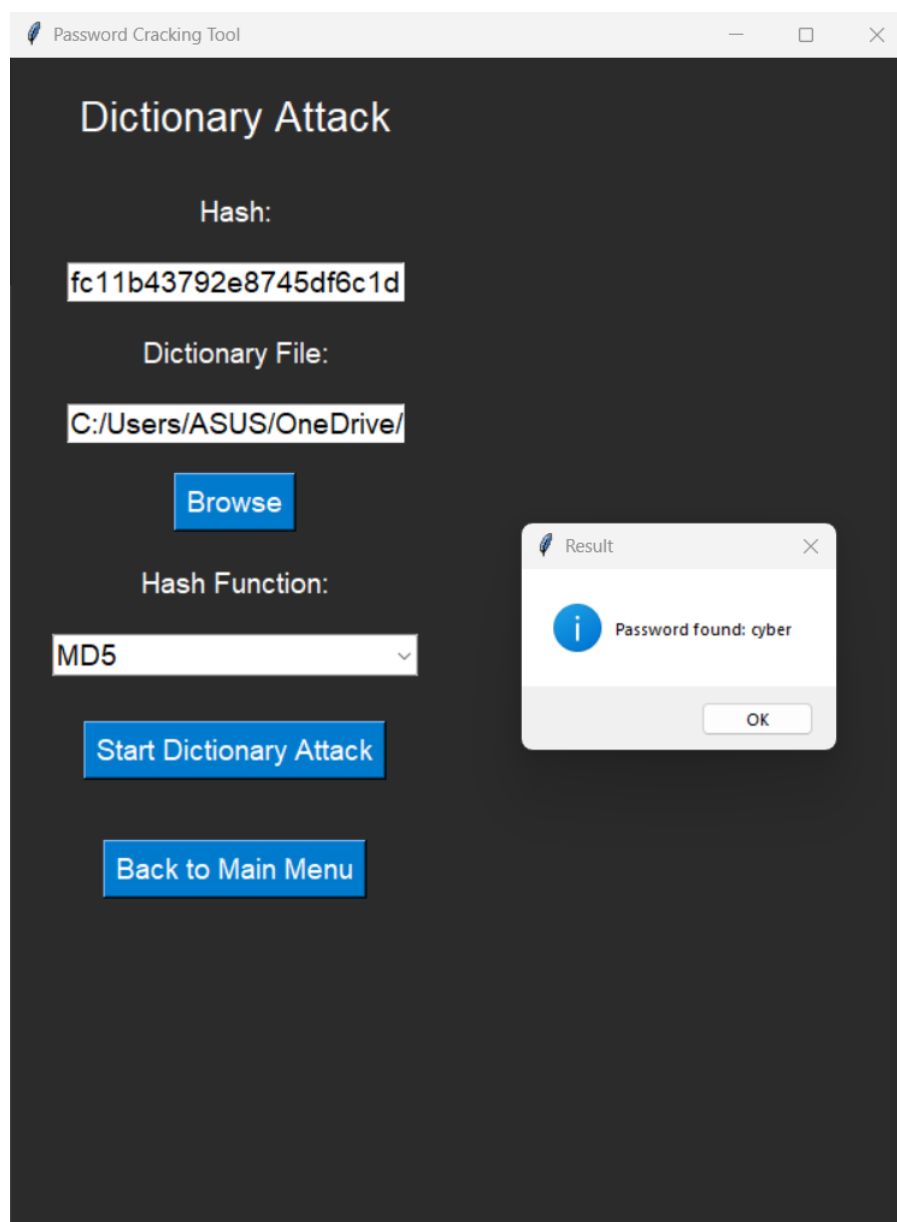


Figure 8: Result of dictionary attack

6.2 Brute Force attack

The Brute Force Attack successfully identified the password by systematically testing all possible combinations within the specified range. Unlike Dictionary Attack, it doesn't rely on a wordlist, making it effective for cracking unknown or customized passwords. The tool displayed the result in a popup after finding the correct match. This highlights the tool's robustness and the importance of using strong, lengthy passwords to resist brute-force attacks.

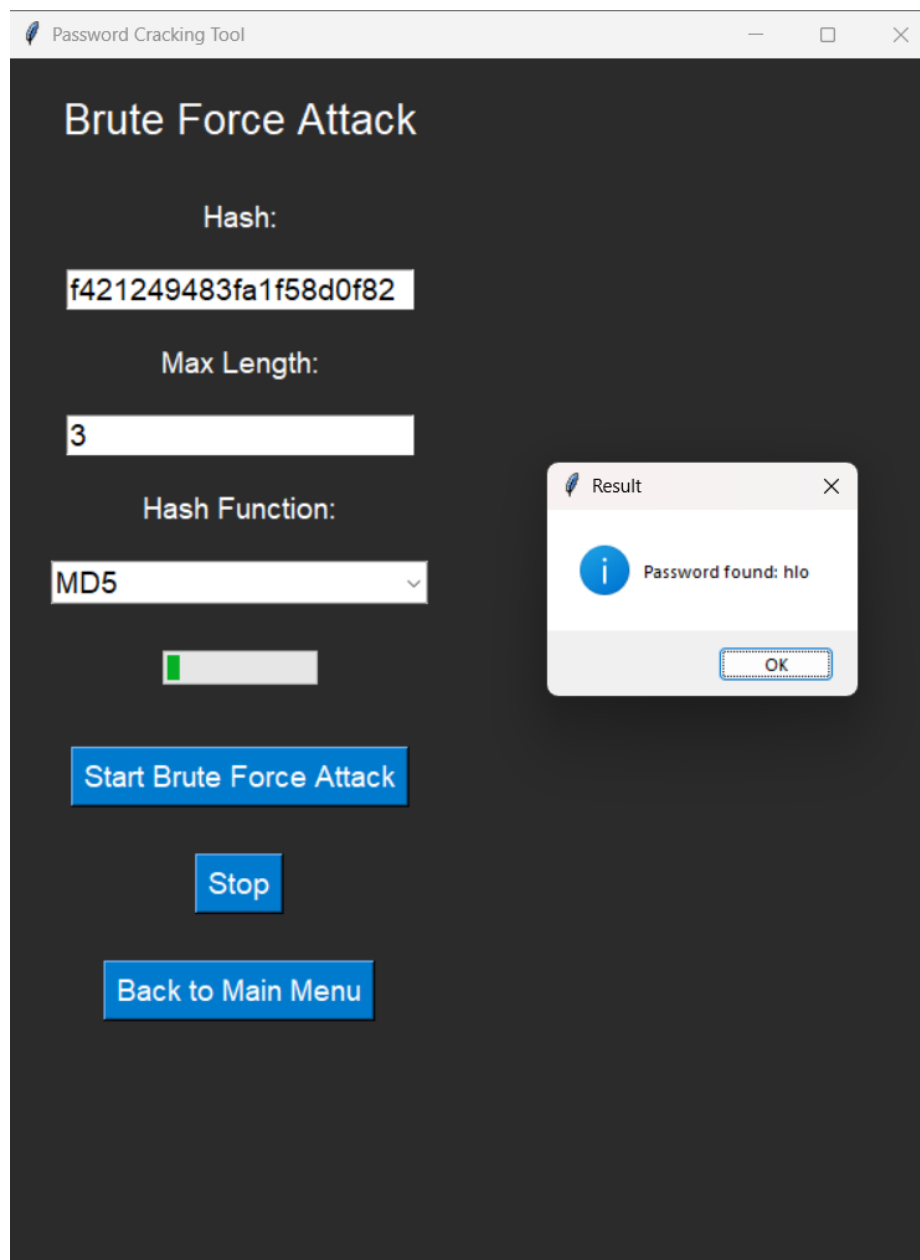


Figure 9: Result of Brute Force Attack

6.3 Wordlist Generator

The Wordlist Generator successfully created a customized list of potential passwords based on the provided character set, minimum length, and maximum length. The generated wordlist is saved to a specified file and is ready for use in password attacks like dictionary or brute force. This feature enables efficient and tailored wordlist creation for various testing scenarios.

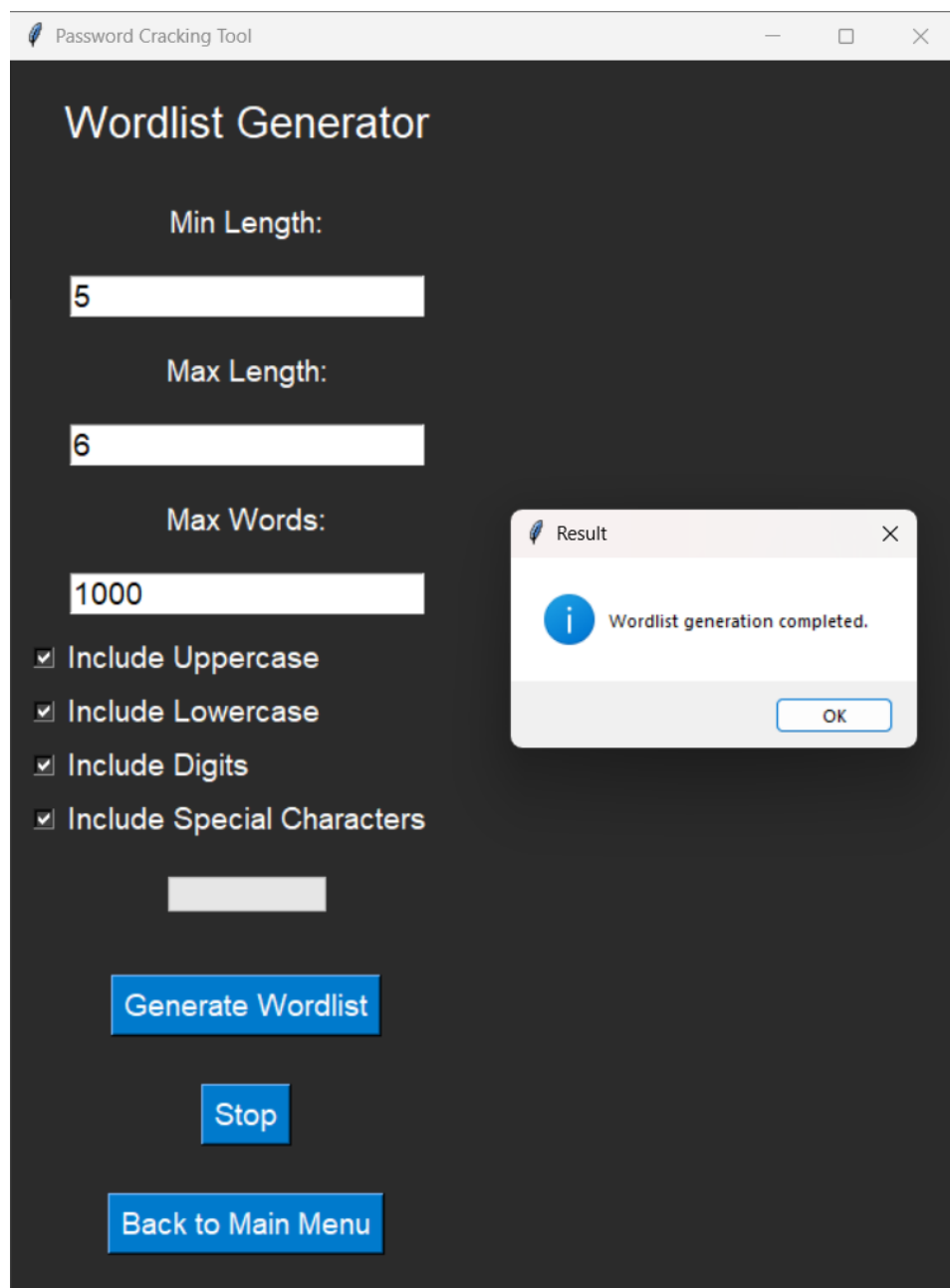
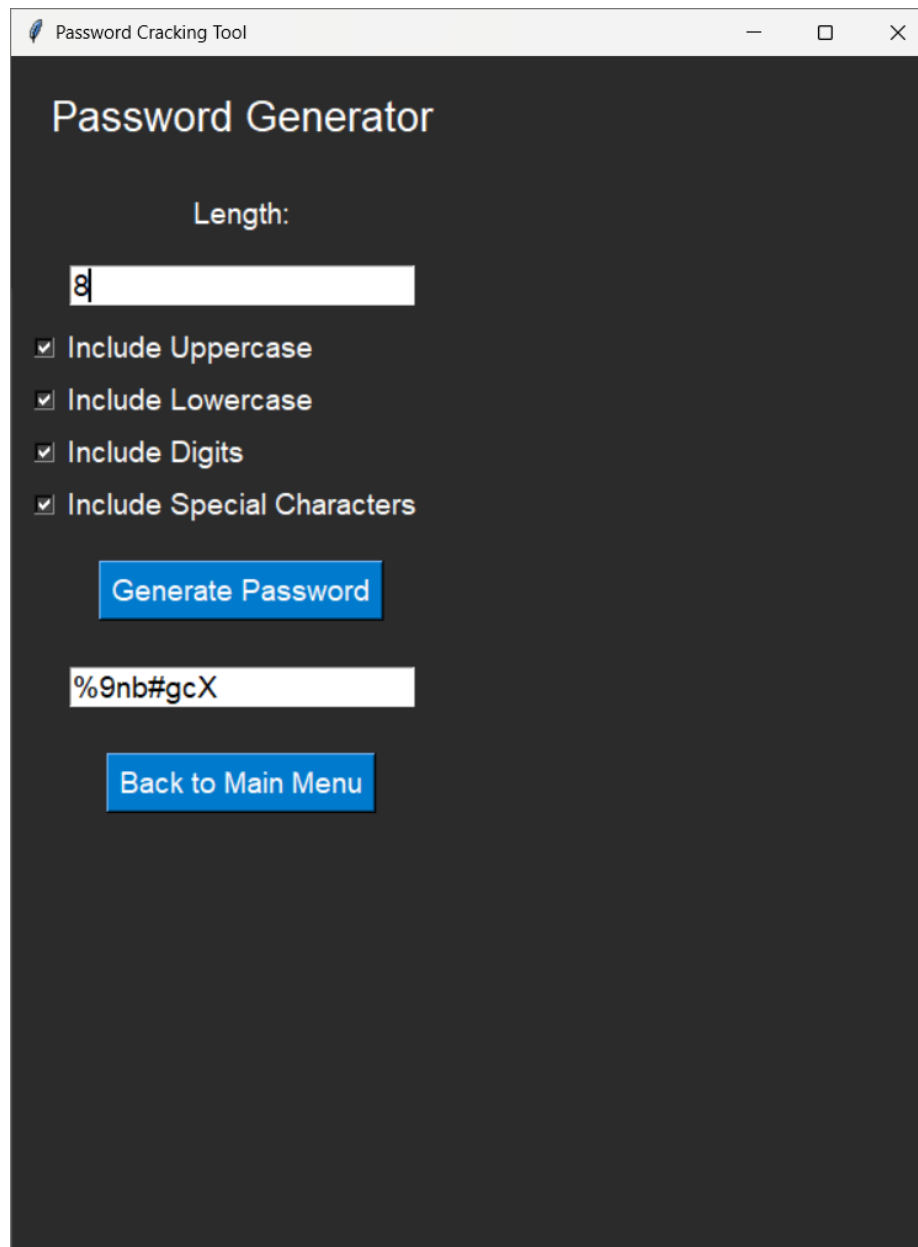


Figure 10: Result of Wordlist Generator

6.4 Password Generator

The Password Generator successfully created strong, random passwords based on the specified criteria, such as length and inclusion of special characters. The generated passwords are displayed for easy use, ensuring enhanced security against cracking attempts. This feature highlights the tool's ability to promote better password practices.



The screenshot shows a window titled "Password Cracking Tool" with a dark background. The main heading is "Password Generator". Below it, the "Length:" is set to "8" in a text input field. There are four checked checkboxes: "Include Uppercase", "Include Lowercase", "Include Digits", and "Include Special Characters". A blue button labeled "Generate Password" is positioned below the checkboxes. Below this button, a text input field displays the generated password "%9nb#gcX". At the bottom, there is a blue button labeled "Back to Main Menu".

Figure 11: Result of Password Generator

6.5 Password Analyzer

The Password Analyzer evaluated the input password and provided a strength rating based on criteria like length, character variety, and complexity. It also offered feedback on improving weak passwords, encouraging stronger security practices. This feature ensures users are aware of their password's vulnerability and helps them enhance protection.

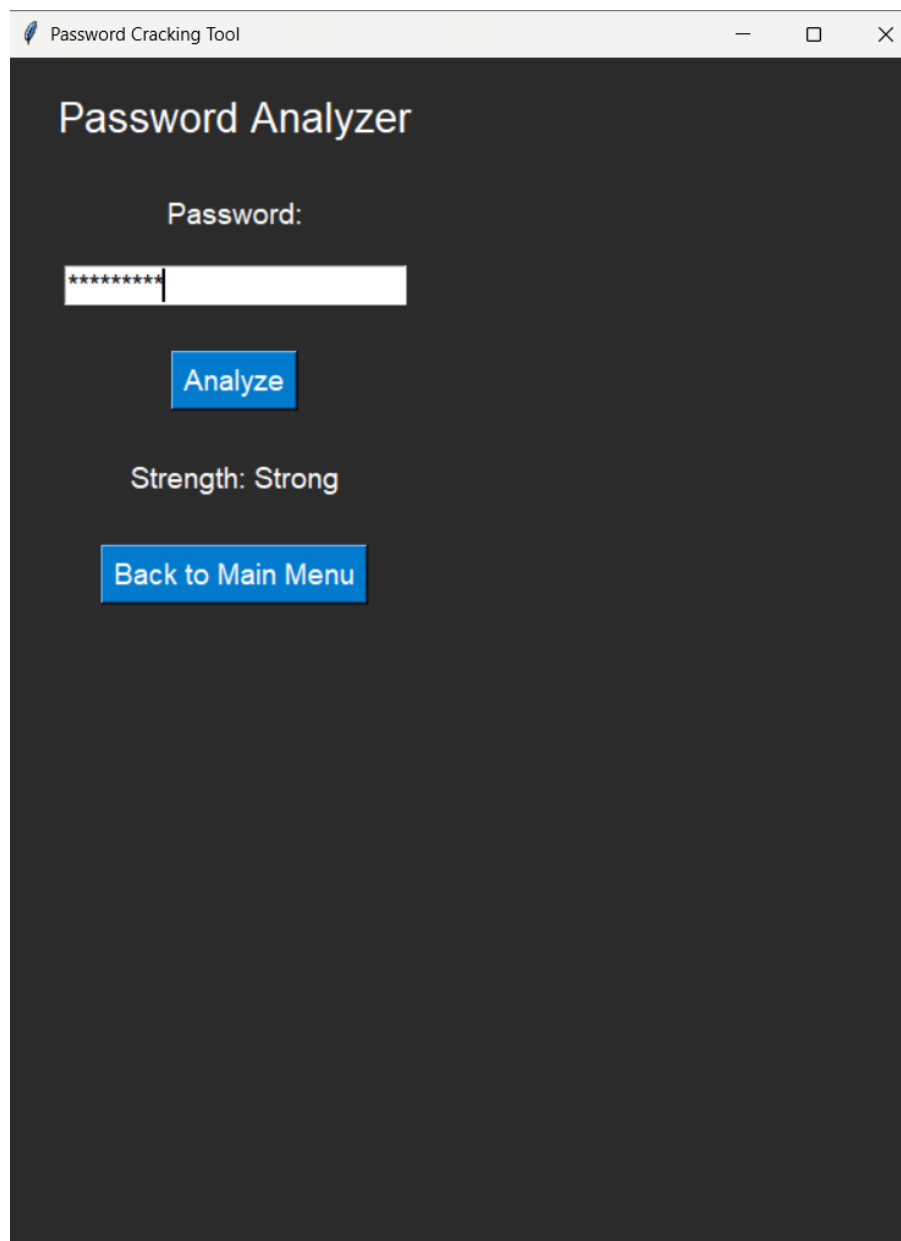


Figure 12: Result of Password Analyzer

CHAPTER 7

CONCLUSION

In conclusion, the Password Cracking and Analysis Tool provides a versatile, user-friendly solution for evaluating password security and conducting hash-cracking operations. It is designed to cater to cybersecurity enthusiasts, researchers, and professionals looking for a tool that combines password strength analysis, cracking techniques, and custom wordlist generation in a single platform. The tool excels in supporting multiple cryptographic hash algorithms, such as MD5, SHA-1, SHA-256, SHA-3, and BLAKE2, and offers advanced attack methods, including Dictionary Attack and Brute Force Attack. Its Password Generator and Password Analyzer modules add value by promoting strong password practices and evaluating password security metrics effectively. Furthermore, the ability to create customized wordlists enhances its usability in real-world scenarios, such as penetration testing and ethical hacking tasks.

Limitations and future enhancements:

The tool, while robust and efficient, has some limitations. Cracking complex passwords or processing extensive wordlists can demand significant computational resources and time. Its support for hash algorithms is limited to widely used types, lacking compatibility with specialized or uncommon algorithms. Additionally, the tool's reliance on user-provided wordlists and parameters may restrict its effectiveness if input data is insufficient or improperly prepared. Moreover, its platform-specific focus ensures seamless operation on Linux but limits portability to other systems. Despite these constraints, the tool's modular design offers flexibility and scalability, making it a strong foundation for future enhancements and broader compatibility.

Potential enhancements for the tool include expanding its capabilities by integrating support for additional hash algorithms, such as crypt, Argon2, and NTLM, to address a wider range of use cases. Performance could be significantly improved through the implementation of parallel processing techniques, including multithreading or GPU acceleration, to expedite brute force and dictionary attacks. Adding real-time hash cracking functionality would enable live monitoring and interception during penetration testing engagements. Extending cross-platform compatibility to Windows and macOS would broaden the tool's accessibility and user base. Finally, enhanced reporting features, such as detailed visual reports, interactive logs, and tailored recommendations, would provide users with deeper insights and actionable feedback.

for improving password security.

The Password Cracking and Analysis Tool is a robust and valuable addition to the arsenal of cybersecurity tools. Its comprehensive feature set, combined with ease of use and modular architecture, makes it a useful tool for password analysis, hash cracking, and educational purposes. By addressing its current limitations and incorporating advanced functionalities, the tool has the potential to become a significant resource in the field of cybersecurity, aiding both defensive and ethical offensive operations.

REFERENCES

1. Password Cracking Technology -
URL:<https://www.researchgate.net/publication/345398833>
Research_on_Password_Cracking_Technology_Based_on_Improved_Transformer.
Accessed Nov 2024.
2. Password Cracking –
URL:<https://www2.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/topic7-final/report.pdf>. Accessed Nov 2024.
3. Overview of password Cracking –URL: <https://ieeexplore.ieee.org/document/7371616>.
Accessed Nov 2024.
4. Brute Force –
URL: <https://ijrpr.com/uploads/V3ISSUE11/IJRPR7767.pdf>. Accessed Nov 2024.
5. GitHub –
URL: <https://github.com/openwall/john>. Accessed Nov 2024.
6. Wordlist Generator –
URL: <https://github.com/J4NN0/wordlist-generator>. Accessed Nov 2024.
7. Password analyzer –
URL: https://www.passwordmonster.com/#google_vignette. Accessed Nov 2024.
8. Cyber Chef –
URL: <https://gchq.github.io/CyberChef/>. Accessed Nov 2024.
9. Password Generator –
URL:<https://www.avast.com/en-in/random-password-generator#pc>.
Accessed Nov 2024.
10. Crack Station –
URL: <https://crackstation.net/>. Accessed Nov 2024.

APPENDIX A: List of figures

Figure Number	Figure Title	Page Number
Figure 1	Example of main window	9
Figure 2	Dictionary Attack	10
Figure 3	Brute force Attack	11
Figure 4	Password Generator	12
Figure 5	Wordlist Generator	13
Figure 6	Password Analyzer	14
Figure 7	Workflow diagram	23
Figure 8	Result of Dictionary Attack	34
Figure 9	Result of Brute Force Attack	35
Figure 10	Result of Wordlist Generator	36
Figure 11	Result of Password Generator	37
Figure 12	Result of Password Analyzer	38

APPENDIX B: Sample Code

1. Dictionary Attack Module

The **Dictionary Attack** module matches a hash against pre-computed passwords from a wordlist.

```
def dictionary attack (self, hash value, hash type, wordlist path):
    try:
        with open (wordlist path, 'r') as wordlist:
            for password in wordlist:
                password = password. Strip ()
                hashed password = highline (hash type, password. Encode ()). hex digest ()
                if hashed password == hash_value:
                    return password found: {password}"
            return "Password not found in the dictionary."
    except Exception as e:
        return ferro during dictionary attack: {e}"
```

- **Functionality:** Reads a wordlist, hashes each password, and compares it with the target hash.
- **Input:** Hash value, hash type (e.g., MD5, SHA-1), and wordlist path.
- **Output:** The cracked password or an error message.

2. Brute Force Attack Module

The **Brute Force Attack** generates and tests all possible password combinations within a given character set.

```
def brute force attack (self, hash value, hash type, charset, min_length, max_length):
    try:
        for length in range (min_length, max_length + 1):
            for candidate in itertools.product(charset, repeat=length):
                candidate = ".join(candidate)
                hashed_candidate = hashlib.new(hash_type, candidate.encode()).hexdigest()
                if hashed_candidate == hash_value:
```

```
        return f"Password found: {candidate}"
    return "Password not found in brute force range."
except Exception as e:
    return f"Error during brute force attack: {e}"
```

- **Functionality:** Iteratively generates password combinations and matches their hashes to the target.
- **Input:** Hash value, hash type, character set, and password length range.
- **Output:** The cracked password or a failure message.

3. Password Generator Module

The **Password Generator** creates strong passwords based on user-defined criteria.

```
def generate_password(self, length, include_uppercase, include_digits, include_special_chars):
    charset = string.ascii_lowercase
    if include_uppercase:
        charset += string.ascii_uppercase
    if include_digits:
        charset += string.digits
    if include_special_chars:
        charset += "!@#$$%^&*()_+~[]{}|;':<.>?/"

    return "".join(random.choice(charset) for _ in range(length))
```

- **Functionality:** Generates random passwords with specified length and character requirements.
- **Input:** Length, whether to include uppercase letters, digits, and special characters.
- **Output:** A randomly generated password.

4. Wordlist Generator Module

The **Wordlist Generator** produces custom wordlists based on input parameters.

```
def generate_wordlist(self, charset, min_length, max_length, output_file):
    try:
        with open(output_file, 'w') as file:
            for length in range(min_length, max_length + 1):
                for word in itertools.product(charset, repeat=length):
```

```
        file.write("".join(word) + '\n')

    return f"Wordlist generated at {output_file}"
except Exception as e:
    return f"Error generating wordlist: {e}"
```

- **Functionality:** Writes all possible combinations of characters within the specified range to a file.
- **Input:** Character set, length range, and output file name.
- **Output:** Generated wordlist file or an error message.

5. Password Analyzer Module

The **Password Analyzer** evaluates a password's strength and provides suggestions for improvement.

```
def analyze_password_strength(self, password):
    issues = []
    strength = "Strong"

    if len(password) < 8:
        strength = "Weak"
        issues.append("Password is too short.")
    if not any(char.isdigit() for char in password):
        strength = "Weak"
        issues.append("Missing numbers.")
    if not any(char.isupper() for char in password):
        strength = "Moderate"
        issues.append("Missing uppercase letters.")
    if not any(char.islower() for char in password):
        strength = "Moderate"
        issues.append("Missing lowercase letters.")
    if not any(char in "!@#%&*()_+~[]{}|;':<>?/" for char in password):
        strength = "Moderate"
        issues.append("Missing special characters.")

    return strength, " | ".join(issues) if issues else "Password meets all requirements."
```


- **Functionality:** Assesses password length, complexity, and entropy.
- **Input:** A password string.
- **Output:** A strength rating (Weak, Moderate, Strong) and improvement suggestions.

6. Report Generation

The tool supports the generation of detailed forensic reports summarizing the results of attacks, password analysis, and other findings.

```
def generate_report(self, filename, data):
    try:
        with open(filename, 'w') as report_file:
            report_file.write(data)
        return f"Report saved to {filename}"
    except Exception as e:
        return f"Error saving report: {e}"
```

- **Functionality:** Writes the analysis and attack results to a file.
- **Input:** File name and data to include in the report.
- **Output:** Confirmation of the saved report or an error message.