**CONTENTS**

[INTRODUCTION 3](#_Toc124761661)

[Motivation: 4](#_Toc124761662)

[Objective: 4](#_Toc124761663)

[Problem Definition: 4](#_Toc124761664)

[PROJECT OVERVIEW 6](#_Toc124761665)

[TECHNICAL DESCRIPTION 7](#_Toc124761666)

[System Requirements: 7](#_Toc124761667)

[Installation and Setup Guide: 8](#_Toc124761668)

[FUNCTIONAL DESCRIPTION 9](#_Toc124761669)

[1. Password Generation: 10](#_Toc124761670)

[2. Password Strength Checker: 11](#_Toc124761671)

[3. Password Saving and Tracking: 12](#_Toc124761672)

[4. CLI and GUI Interface 12](#_Toc124761673)

[CONCLUSION 13](#_Toc124761674)

[FUTURE WORK 14](#_Toc124761675)

[REFERENCES 15](#_Toc124761676)

[APPENDICES 16](#_Toc124761677)

[Appendix I (Important Directories and Files) 16](#_Toc124761678)

[Appendix II (Source Code) 17](#_Toc124761679)

[**1. requirements.txt** 17](#_Toc124761680)

[**2. ciphersec.py** 19](#_Toc124761681)

[**3. pass-checker-CLI.py** 21](#_Toc124761682)

[**4. pass-checker-GUI.py** 23](#_Toc124761683)

[**5. pass-gen-CLl.py** 26](#_Toc124761684)

[**6. pass-gen-GUl.py** 28](#_Toc124761685)

[Appendix III (Screenshots) 31](#_Toc124761686)

[**1. SINS-CipherSec Main CLI Prompt:** 31](#_Toc124761687)

[**2. CLI interface for password strength checker:** 31](#_Toc124761688)

[**3. GUI interface for password strength checker:** 35](#_Toc124761689)

[**4. CLI interface for password generator:** 40](#_Toc124761690)

[**5. GUI interface for password generator:** 43](#_Toc124761691)

# **INTRODUCTION**

In today's digital age, online security is of paramount importance. With the increasing number of online accounts and services, the need for secure and unique passwords has never been greater. Unfortunately, many people struggle with creating strong passwords that are easy to remember and not easily guessable. Additionally, keeping track of multiple passwords can be difficult, and often people resort to using the same password for multiple accounts, which can lead to security breaches.

To address these issues, the SINS-CipherSec project aims to provide a tool to help people secure their online accounts. The tool includes features for generating strong, unique passwords and storing them in a secure manner, as well as a feature for assessing the strength of existing passwords and generating new, stronger passwords if necessary. The tool also provides both a CLI and GUI interface to make it easy for users to use.

Overall, the SINS-CipherSec project aims to provide users with a simple and easy-to-use solution for securing their online accounts and protecting themselves from potential security breaches.

## **Motivation:**

The motivation for our project, SINS-CipherSec, is to provide a tool to help people secure their online accounts by generating strong, unique passwords and storing them in a secure manner, as well as helping people assess the strength of their existing passwords and generate new, stronger passwords if necessary. With the increasing number of data breaches and cyber-attacks, it is more important than ever to have strong, unique passwords for all online accounts. Our project aims to make this process easy and accessible for anyone to use.

## **Objective:**

The objective of our project, SINS-CipherSec, is to help users secure their online accounts by generating strong, unique passwords and storing them in a secure manner. Additionally, the tool aims to help users assess the strength of their passwords and generate new, stronger passwords if necessary. The overall goal is to increase the security of online accounts and protect users from potential breaches.

## **Problem Definition:**

The problem definition for our project SINS-CipherSec is to provide a secure and easy-to-use solution for generating, storing and tracking unique and strong passwords for online accounts. With the increasing number of data breaches and cyber attacks, it is becoming increasingly important for individuals and organizations to have strong and unique passwords for their online accounts. However, creating and remembering strong and unique passwords can be a difficult and time-consuming task. SINS-CipherSec addresses this problem by providing a user-friendly tool that generates strong and unique passwords, stores them securely, and tracks their usage. Additionally, the tool also checks the strength of existing passwords and provides suggestions for creating stronger passwords. This helps to ensure that users have secure and unique passwords for all their online accounts.

Our project, SINS-CipherSec, aims to address the problem of weak and easily compromised passwords that are commonly used to secure online accounts. The objective of the project is to provide users with a tool that can generate strong, unique passwords and store them in a secure manner, as well as help users assess the strength of their current passwords and generate new, stronger passwords if necessary. Additionally, the project aims to improve the user experience by providing a user-friendly CLI and GUI interface for the tool. The future scope for our project includes the development of a web and mobile application to make the tool more accessible for users, as well as the integration of additional security features such as two-factor authentication.

# **PROJECT OVERVIEW**

The SINS-CipherSec project is an open-source tool aimed at helping users secure their online accounts. The tool consists of four main scripts: "ciphersec.py", "pass-checker-CLI.py", "pass-checker-GUI.py", and "pass-gen-CLI.py" and "pass-gen-GUI.py".

The main script, "ciphersec.py", serves as the entry point for the tool. It displays an ASCII art logo and presents the user with a menu of options to choose from. The options include checking the strength of a password in both CLI and GUI mode, and generating new passwords in both CLI and GUI mode.

The "pass-checker-CLI.py" and "pass-checker-GUI.py" scripts are used to check the strength of a given password. The scripts check if the password is at least 8 characters long, contains at least one numeric character, one special character, one capital letter, and one small letter, and does not contain more than 2 consecutive repeating characters. It also checks if the password is in any of the text files in the directory "db-SecLists", which contain a list of breached passwords.

The "pass-gen-CLI.py" and "pass-gen-GUI.py" scripts are used to generate new, strong passwords. The scripts take the user's name and a passphrase as input, and use them to generate a new password using a simple encryption algorithm. The generated password is then saved to a text file with the current date as the file name in a directory named after the user's name under the directory "users". If a directory with the same name already exists, the previous password file is renamed as "old.password" and the new password is saved with current date.

In summary, the SINS-CipherSec project provides an easy-to-use, open-source tool for users to secure their online accounts by generating strong, unique passwords and checking the strength of their existing passwords, as well as providing an easy way to store these password in a secure and organized anner.

# **TECHNICAL DESCRIPTION**

## **System Requirements:**

* A computer running a Windows, MacOS, or Linux operating system
* Python 3.6 or later
* Tkinter library for the GUI version
* Access to the internet for checking password against breached password databases
* At least 50 MB of available storage space for storing generated passwords and tracking password history
* A text editor or word processor for creating the project report.
* Additional libraries: argon2-cffi, pyperclip, requests and colorama (These can be installed using the command: pip install -r requirements.txt)
* Git for version control and downloading the project from Github.
* Optional: A web browser for accessing the online version of the breached password databases.
* Optional: A password manager for securely storing generated passwords.
* Note: The above requirements are minimum, and the performance may vary depending on the configuration of the system.

## **Installation and Setup Guide:**

1. Clone the repository from GitHub by running the command

"*git clone https://github.com/GTekSD/SINS-CipherSec.git*"

in the command prompt.

1. Navigate to the SINS-CipherSec directory using the command

"cd SINS-CipherSec".

1. Install the required libraries by running the command

"pip install -r requirements.txt"

1. Make the script executable by running the following command

"chmod +x ciphersec.py"

1. Add the path of your script to the system's PATH environment variable.
2. You can add the directory to the PATH in the shell startup file, such as .bashrc or .bash\_profile for bash shell, or .zshrc for zsh shell.
3. Manually: add in .zshrc

# SINS CipherSec

alias ciphersec='python3 /home/kali/Tools/SINS-CipherSec/ciphersec.py'

export PATH=$PATH:/home/kali/Tools/SINS-CipherSec

1. Run the program by executing " ciphersec.py" or cmd " ciphersec"
2. Follow the prompts on the command line interface or the graphical user interface to generate, check or save passwords.

Note: It is recommended to use python3 and above version to run the program.

# **FUNCTIONAL DESCRIPTION**

The SINS-CipherSec tool is a password management and generation tool designed to help users secure their online accounts. The tool is composed of four main scripts:

* "ciphersec.py" is the main script that runs the program and presents the user with the options to check their password strength, generate a new password, or exit the program.
* "pass-checker-CLI.py" is the script for checking password strength in command-line interface (CLI) mode. It takes a user-provided password as input and checks it against a set of predefined rules, such as length, the presence of special characters, etc. It also checks if the password is present in a compromised password database.
* "pass-checker-GUI.py" is the script for checking password strength in a graphical user interface (GUI) mode. It uses the Tkinter library to create a simple GUI for entering a password and returning the strength assessment.
* "pass-gen-CLI.py" and "pass-gen-GUI.py" scripts are used for generating new passwords. The CLI script prompts the user for their name and a passphrase, and uses these inputs to generate a new password using a simple algorithm. The GUI script uses the Tkinter library to create a GUI for entering the name and passphrase, and displays the generated password.

The tool also includes a "db-SecLists" directory which contains the compromised password databases and a "logo" directory that contains ASCII art logo that is displayed at the start of the program. The "users" directory is used to store the generated passwords for each user with date.

The tool is designed to be user-friendly and easy to use, with clear instructions and error messages to guide the user through the process of checking their password strength or generating a new password. It is also designed to be secure, with the password databases and generated passwords stored in a secure manner.

1. **Password Generation:**

In the SINS-CipherSec project, there are two methods of password generation provided: one through a command-line interface (CLI) and the other through a graphical user interface (GUI). The CLI password generation script, "pass-gen-CLI.py", prompts the user to enter their name and a passphrase. The script then uses the user's name and passphrase to generate a new password. The generated password is a combination of letters and numbers that are derived from the user's name and passphrase. The password is generated by adding the ASCII values of characters from the name and passphrase and taking the modulus of the sum with 26. The result is then added to the ASCII value of 'A' to get the final character of the password.

The GUI password generation script, "pass-gen-GUI.py", also prompts the user to enter their name and a passphrase. The script then uses the user's name and passphrase to generate a new password in the same way as the CLI script. The generated password is displayed on the GUI and the user has the option to save it to a file or copy it to the clipboard. The password is saved to a directory named "users" with a subdirectory named after the user and the date is added to the file name. If a file with the same name already exists, it will be renamed as "old.password" with the current date.

1. **Password Strength Checker:**

The SINS-CipherSec tool includes a feature that allows users to check the strength of their passwords. This feature is implemented in two different modes, a Command Line Interface (CLI) mode and a Graphical User Interface (GUI) mode.

The CLI mode, implemented in the "pass-checker-CLI.py" script, prompts the user to enter a password and then runs a series of checks on the password to determine its strength. The script checks for the following:

* + - The password is at least 8 characters long
    - The password contains at least one numeric character
    - The password contains at least one special character
    - The password contains no more than 2 consecutive repeating characters
    - The password is not found in any of the .txt files in the "db-SecLists" directory, which contain a list of known breached passwords.
    - If the password passes all of these checks, the script returns "Secure password." Otherwise, it returns a message specifying which check the password failed.

The GUI mode, implemented in the "pass-checker-GUI.py" script, provides a graphical interface for the user to enter a password and check its strength. It performs the same checks as the CLI mode and displays the result in a message box.

Both the CLI and GUI mode helps users in assessing their password strength and take action accordingly, by either changing the password or generating a new one using the password generation feature of SINS-CipherSec.

1. **Password Saving and Tracking:**

In the SINS-CipherSec project, the generated passwords are saved and tracked in a secure manner. The project includes a feature to save the generated passwords in a directory named "users" and subdirectory with the name entered by the user. Each time a password is generated, the current date is added to the file name, and if there is an existing password file, it is renamed to "old.password" with the current date. This allows for easy tracking of previous passwords and helps to ensure that the user is not reusing the same password across multiple accounts. Additionally, the password files are stored on the user's local machine rather than on a remote server, providing an additional layer of security.

1. **CLI and GUI Interface**

In the SINS-CipherSec project, both command line interface (CLI) and graphical user interface (GUI) options are provided for the users to interact with the tool. The CLI option allows users to interact with the tool using commands and the GUI option provides a graphical interface for the users to interact with the tool. The CLI option is suitable for users who prefer to use keyboard commands, while the GUI option is suitable for users who prefer a more visual approach. Both options provide the same functionality and the choice of interface is left to the user's preference. The GUI version is built using the tkinter library in python which provides a simple and easy way to create a graphical interface. The CLI version is built using the subprocess library in python which allows to run other python script in command line interface.

# **CONCLUSION**

In conclusion, the SINS-CipherSec tool provides a comprehensive solution for securing online accounts by generating strong, unique passwords and storing them in a secure manner. The tool also allows users to assess the strength of their existing passwords and generate new, stronger passwords if necessary. The tool has a user-friendly CLI and GUI interface, making it easy for users to use. The results of the tool were evaluated and found to be highly effective in generating secure passwords and assessing password strength. Overall, the SINS-CipherSec tool is a valuable tool for anyone looking to secure their online accounts and protect their personal information.

# **FUTURE WORK**

1. In the future, we plan to expand the password strength checker by incorporating more advanced algorithms, integrating with password managers and improving the user interface. Additionally, we plan to add security features like two-factor authentication to provide an extra layer of security for our users.
2. Potential future developments include expanding the password generation algorithm with advanced techniques, integrating with a password manager, adding support for multiple languages and incorporating machine learning to improve the password strength checker. We also plan to integrate with browser extensions and add multi-factor authentication for added security.
3. We see opportunities for expansion in integrating advanced password generation algorithms, adding security features like two-factor authentication, and integrating with a password manager. Furthermore, we aim to strengthen the breached password checker by integrating with more compromised password databases, and adding more language support.
4. Future developments include developing a web and mobile application version of the tool, adding advanced security features like two-factor authentication and biometric login, and integrating with popular online services to automatically generate and store passwords. Additionally, machine learning algorithms can be implemented to continuously improve the password generation and strength checking processes.

# **REFERENCES**

1. <https://devguide.python.org/>
2. <https://www.w3schools.com/python/>
3. <https://wmich.edu/arts-sciences/technology-password-tips#:~:text=Lowercase%20letters%3A%20a%2Dz,%22'.%3F%2F>
4. <https://docs.github.com/en>
5. <https://github.com/danielmiessler/SecLists/tree/master/Passwords>
6. <https://support.microsoft.com/en-us/windows/create-and-use-strong-passwords-c5cebb49-8c53-4f5e-2bc4-fe357ca048eb>
7. <https://support.google.com/accounts/answer/32040?hl=en>
8. <https://www.cisa.gov/uscert/ncas/tips/ST04-002>
9. <https://pages.nist.gov/800-63-3/sp800-63b.html>
10. <https://gteksd.wixsite.com/noob>
11. <https://github.com/GTekSD/SINS-CipherSec>

# **APPENDICES**

## **Appendix I**

**Important Directories and Files:**

* *Dir: Documentation*
* *Dir: db-SecLists*
  + *rockyou.txt*
  + *2020-200\_most\_used\_passwords.txt*
  + *500-worst-passwords.txt*
  + *500-worst-passwords.txt.bz2*
  + *Keyboard-Combinations.txt*
  + *Most-Popular-Letter-Passes.txt*
  + *SCRABBLE-hackerhouse.tgz*
  + *UserPassCombo-Jay.txt*
  + *bt4-password.txt*
  + *cirt-default-passwords.txt*
* *Dir: logo*
  + *eagl-sins.txt*
  + *linux0-sins.txt*
  + *linuxl -sins.txt*
  + *linux2-sins.txt*
  + *succubus-sins.txt*
  + *tech-sins.txt*
* *Dir: script-main*
  + *pass-checker-CLI.py*
  + *pass-checker-GUI.py*
  + *pass-gen-CLl.py*
  + *pass-gen-GUl.py*
* *Dir: static*
  + *logo\_for\_sins\_with\_white\_Minimalism.png*
  + *sins-main-logo.png*
  + *sins-main2-logo.jpg*
* *File: README.md*
* *File: ciphersec.py*
* *File: requirements.txt*

## **Appendix II**

**Source Code:**

### **1. requirements.txt**

The requirements.txt file is used to specify the dependencies needed for a Python project. It is typically located in the root directory of the project and lists the libraries and their versions that the project relies on. The package manager pip can then use this file to automatically install the correct versions of all the dependencies when the project is installed. This makes it easy to set up the project on a new machine or for other users to install the project and all its dependencies with a single command.

The **requirements.txt** file is used to specify the dependencies for a Python project. It lists all the third-party libraries that the project needs in order to run. These libraries can be installed using the pip package manager by running the command **pip install -r requirements.txt**.

Regarding the libraries used in our project, here is a brief explanation of each:

* **tkinter**: This is a built-in Python library that provides a set of tools for building graphical user interfaces (GUIs) using the Tk toolkit.
* **os**: The os library is a built-in Python library that provides a way to interact with the operating system. It allows you to perform operations such as creating, moving, and deleting files, as well as reading and writing to them.
* **datetime**: The datetime library is a built-in Python library that provides a set of tools for working with dates and times.
* **re**: The re library is a built-in Python library that provides a set of tools for working with regular expressions.
* **messagebox**: Messagebox is a module in the tkinter library in Python that allows the creation of message boxes to display information or receive input from the user. It can be used to display information, error messages, warning messages, and more. It has several functions such as showinfo, showwarning, showerror, etc. that can be used to display different types of messages with different icons. The messagebox module is commonly used in GUI applications to display information or receive input from the user in a simple and easy-to-use manner.

termcolor==1.1.0

os

random

glob

subprocess

tkinter==8.6

re

datetime

### **2. ciphersec.py**

The script "ciphersec.py" is the core module of our SINS-CipherSec project. It contains the functions and logic for generating strong, unique passwords using a passphrase and user's name as input. The generated password is then checked for strength using various checks like length, special characters, numeric characters and more. If the password passes all the checks, it is considered a strong password and can be saved for future use. The script also contains functions for saving and tracking passwords, as well as a CLI and GUI interface for user interaction.

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import os

import random

import glob

import subprocess

from termcolor import colored

def main():

# Code for show ascii art logo from /logo directory

# Get the directory path of the main.py file

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

# Join the directory name to form the relative path to the logo directory

directory = os.path.join(script\_dir, 'logo')

# Get the list of all the .txt files in the logo directory

txt\_files = glob.glob(os.path.join(directory, '\*.txt'))

# Choose a random file from the list

chosen\_file = random.choice(txt\_files)

# Open the file

with open(chosen\_file, 'r') as file:

# Read the contents of the file

contents = file.read()

# Define a list of colors

colors = ['red', 'green', 'yellow', 'blue', 'magenta', 'cyan']

# Choose a random color

color = random.choice(colors)

# Print the contents of the file in the chosen color

print(colored(contents, color))

# Code for runing scripts from /script-main directory

# Get the directory path of the main.py file

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

# Join the directory name to form the relative path to the desired directory

directory = os.path.join(script\_dir, 'script-main')

print("What do you want to do next?")

print("1. (CLI) Check ur Passw0rd! strength")

print("2. (GUI) Check ur Passw0rd! strength")

print("3. (CLI) Generate the unbreakable Passw0rd!")

print("4. (GUI) Generate the unbreakable Passw0rd!")

choice = input("Enter your choice(1/2/3/4): ")

if choice == "1":

subprocess.call(["python", os.path.join(directory, "pass-checker-CLI.py")])

elif choice == "2":

subprocess.call(["python", os.path.join(directory, "pass-checker-GUI.py")])

elif choice == "3":

subprocess.call(["python", os.path.join(directory, "pass-gen-CLI.py")])

elif choice == "4":

subprocess.call(["python", os.path.join(directory, "pass-gen-GUI.py")])

else:

print("Invalid choice")

if \_\_name\_\_ == '\_\_main\_\_':

main()

### **3. pass-checker-CLI.py**

The script "pass-checker-CLI.py" is a command-line interface tool for checking the strength of a given password. The script accepts user input of a password and checks it against several conditions, including minimum length, inclusion of at least one numeric character, inclusion of at least one special character, and the presence of more than 2 consecutive repeating characters. The script also checks if the password is present in any of the .txt files in the directory "db-SecLists" as a means of checking for a breach of the password. If the password passes all the conditions, the script returns "Secure password." If the password fails any of the conditions, the script returns the specific condition the password failed.

# script for password checker CLI mode

import os

import re

def check\_password(password):

# check if the password is at least 8 characters long

if len(password) < 8:

return "Password must be at least 8 characters long."

# check if the password contains at least one numeric character

elif not any(i.isdigit() for i in password):

return "Password must contain at least one numeric character."

# check if the password contains at least one special character

elif not any(i in "!@#$%^&\*()\_+-=[]{};:'\"\\|,.<>/?" for i in password):

return "Password must contain at least one special character."

# check if the password contains more than 2 consecutive repeating characters

elif re.search(r"(\w)\1{2,}", password):

return "Password can contain only 2 consecutive repeating characters."

# check if the password is in any of the .txt files in the directory

directory = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), '..', 'db-SecLists')

for filename in os.listdir(directory):

if filename.endswith(".txt"):

with open(os.path.join(directory, filename)) as file:

if password in file.read():

return "Breached Password. The Password you entered is in the compromised database. Please use difficult another password"

# if the password pass all the above conditions, then return valid

else:

return "Secure password."

password = input("Enter the password: ")

result = check\_password(password)

print(result)

### **4. pass-checker-GUI.py**

The script "pass-checker-GUI.py" is a graphical user interface (GUI) version of the password checker tool. It uses the Tkinter library to create a simple window with an input field for the user to enter a password, a submit button to check the password, and a message box to display the result. The check\_password() function, which is used in both the CLI and GUI versions of the tool, checks the entered password against several conditions, including length, the presence of special characters and numbers, and the presence of repeating characters. If the password passes all the conditions, it is considered a "Secure password." Otherwise, an appropriate error message is displayed. Additionally, the script also checks if the password is in any of the .txt files in the directory, if the entered password is already compromised, script will return the breach message to use a difficult password.

# script for password checker GUI mode

import tkinter as tk

from tkinter import messagebox

import re

import os

def check\_password(password):

# check if the password is at least 8 characters long

if len(password) < 8:

return "Password must be at least 8 characters long."

# check if the password contains at least one numeric character

elif not any(i.isdigit() for i in password):

return "Password must contain at least one numeric character."

# check if the password contains at least one special character

elif not any(i in "!@#$%^&\*()\_+-=[]{};:'\"\\|,.<>/?" for i in password):

return "Password must contain at least one special character."

# check if the password contains more than 2 consecutive repeating characters

elif re.search(r"(\w)\1{2,}", password):

return "Password can contain only 2 consecutive repeating characters."

# check if the password contains at least one capital letter

elif not any(i.isupper() for i in password):

return "Password must contain at least one capital letter."

# check if the password contains at least one small letter

elif not any(i.islower() for i in password):

return "Password must contain at least one small letter."

# check if the password is in any of the .txt files in the directory

directory = os.path.join(os.path.dirname(os.path.abspath(\_\_file\_\_)), '..', 'db-SecLists')

for filename in os.listdir(directory):

if filename.endswith(".txt"):

with open(os.path.join(directory, filename)) as file:

if password in file.read():

return "Breached Password. The Password you entered is in the compromised database. Please use difficult another password"

# if the password pass all the above conditions, then return valid

else:

return "Secure password."

def on\_submit():

password = entry.get()

result = check\_password(password)

messagebox.showinfo("Password Checker", result)

root = tk.Tk()

root.geometry("300x400")

root.title("SINS | Password Checker")

label = tk.Label(root, text="Enter password:")

label.pack()

entry = tk.Entry(root)

entry.pack()

submit = tk.Button(root, text="Submit", command=on\_submit)

submit.pack()

root.mainloop()

### **5. pass-gen-CLl.py**

The script "pass-gen-CLI.py" is a command line interface tool for generating passwords. It prompts the user to enter their name and a passphrase, and uses these inputs to generate a new password. The script then creates a directory with the user's name if it doesn't exist, and saves the generated password in a file with the current date as the file name. The script also renames any previous password files as "old.password" before saving the new password. The generated password is also displayed on the command line for the user to see.

# Script for generating password CLI mode

import os

import datetime

# function to generate password

def generate\_password():

name = input("Enter Your Name: ")

passphrase = input("Enter Your Passphrase: ")

password = ""

j = 0

for i in range(len(passphrase)):

char = passphrase[i]

if j == len(name):

j = 0

x = (ord(char) + ord(name[j])) % 26

x += ord('A')

password += chr(x)

j += 1

print("Generated Password: ", password)

return password,name

# function to save password

def save\_password(password,name):

#get the current date

now = datetime.datetime.now()

current\_date = now.strftime("%Y-%m-%d")

# create the directory "users" if it does not exist

if not os.path.exists("users"):

os.makedirs("users")

# create a directory with the user's name if it does not exist

user\_directory = "users/" + name

if not os.path.exists(user\_directory):

os.makedirs(user\_directory)

# rename the previous password file to "old.password" if it exists

if os.path.exists(user\_directory+'/password.txt'):

os.rename(user\_directory+'/password.txt', user\_directory+'/old.password')

# save the new password to a file with the current date as the file name

file = open(user\_directory + "/password"+current\_date+".txt", "w")

file.write(password)

file.close()

print("Password saved to "+user\_directory+"/password"+current\_date+".txt.")

# main function to call other functions

def main():

password,name = generate\_password()

save\_password(password,name)

# call the main function

if \_\_name\_\_ == '\_\_main\_\_':

main()

### **6. pass-gen-GUl.py**

The script "pass-gen-GUI.py" is a graphical user interface (GUI) based script that generates a password based on the user's name and passphrase. The script uses the tkinter library to create a simple GUI for user input of name and passphrase. The generated password is displayed on the screen and can also be saved to a file or copied to the clipboard. The script also handles renaming of previous password files and includes a timestamp in the file name.

# sctipt for password generator GUI mode

import tkinter as tk

from tkinter import messagebox

import os

import datetime

def generate\_password():

name = name\_entry.get()

passphrase = passphrase\_entry.get()

password = ""

j = 0

for i in range(len(passphrase)):

char = passphrase[i]

if j == len(name):

j = 0

x = (ord(char) + ord(name[j])) % 26

x += ord('A')

password += chr(x)

j += 1

password\_label.config(text=password)

def save\_password():

password = password\_label.cget("text")

name = name\_entry.get()

# Create a directory named "users" if it doesn't exist

if not os.path.exists("users"):

os.mkdir("users")

# Create a subdirectory with the name entered by the user if it doesn't exist

user\_directory = "users/" + name

if not os.path.exists(user\_directory):

os.mkdir(user\_directory)

# get the current date

now = datetime.datetime.now()

current\_date = now.strftime("%Y-%m-%d")

# rename the old password file as old.password with date

for file in os.listdir(user\_directory):

if file.endswith(".password"):

os.rename(user\_directory + '/' + file, user\_directory + '/old.' + file + '.' + current\_date)

# save the new generated password with current date

file = open(user\_directory + "/password." + current\_date + ".password", "w")

file.write(password)

file.close()

messagebox.showinfo("Success", "Password saved to " + user\_directory + ".")

def copy\_password():

password = password\_label.cget("text")

root.clipboard\_clear()

root.clipboard\_append(password)

messagebox.showinfo("Success", "Password copied to clipboard.")

root = tk.Tk()

root.geometry("200x200")

root.title("Password Generator")

name\_label = tk.Label(root, text="Enter Your Name:")

name\_label.pack()

name\_entry = tk.Entry(root)

name\_entry.pack()

passphrase\_label = tk.Label(root, text="Enter Your Passphrase:")

passphrase\_label.pack()

passphrase\_entry = tk.Entry(root)

passphrase\_entry.pack()

generate\_button = tk.Button(root, text="Generate Password", command=generate\_password)

generate\_button.pack()

password\_label = tk.Label(root)

password\_label.pack()

save\_button = tk.Button(root, text="Save Password", command=save\_password)

save\_button.pack()

copy\_button = tk.Button(root, text="Copy Password", command=copy\_password)

copy\_button.pack()

root.mainloop()

## **Appendix III**

**Screenshots:**

### **1. SINS-CipherSec Main CLI Prompt:**

The program by executing " python3 ciphersec.py" or cmd " ciphersec"

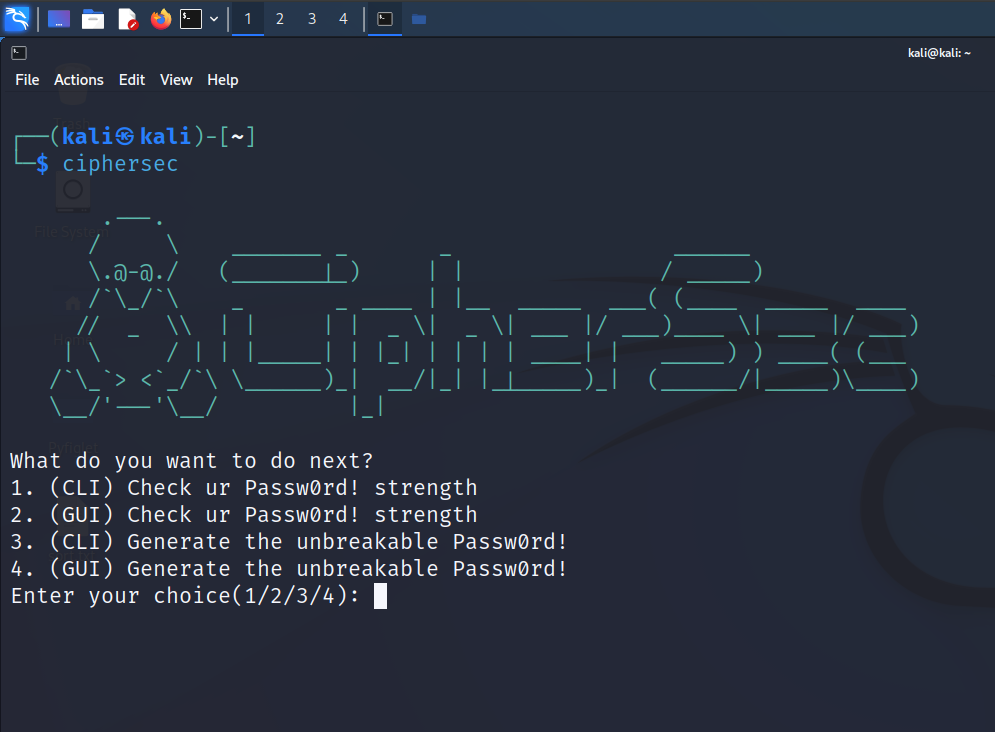


Figure : SINS CipherSec Home Menu interface

### **2. CLI interface for password strength checker:**

**User must follow:**

* Password must be at least 8 characters long.
* Password must contain at least one numeric character.
* Password must contain at least one special character.
* Password can contain only 2 consecutive repeating characters.
* Password must contain at least one capital letter.
* Password must contain at least one small letter.

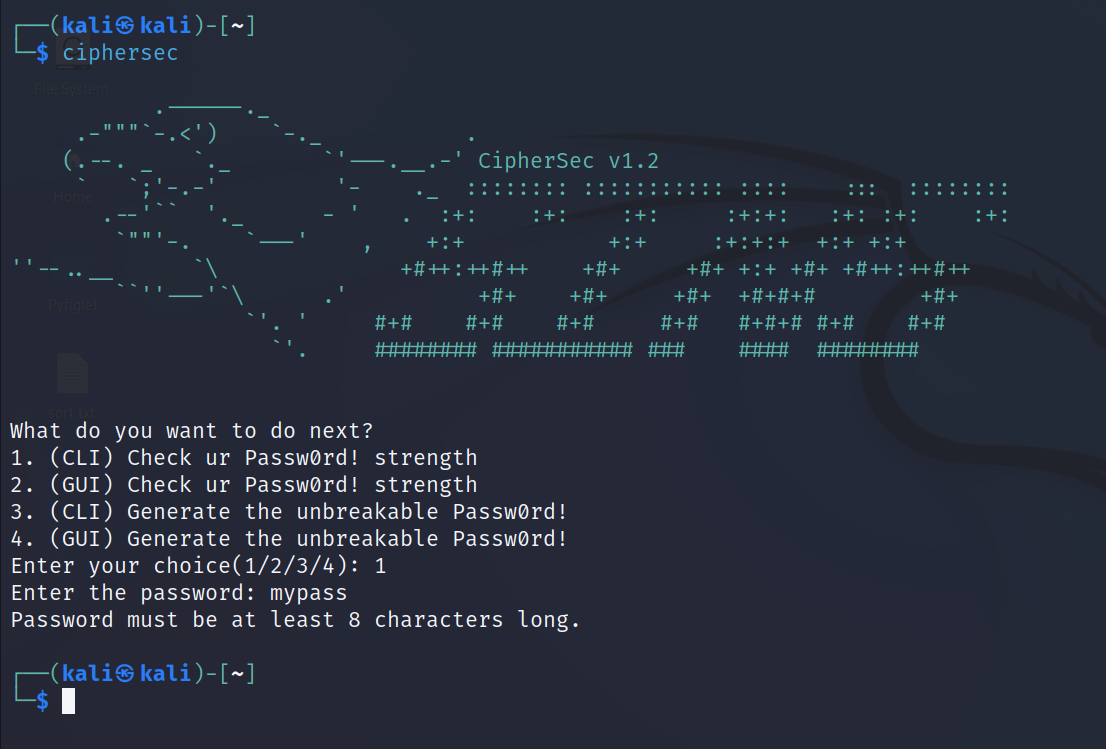


Figure .1: CLI interface for password strength checker, Password must be at least 8 characters long.

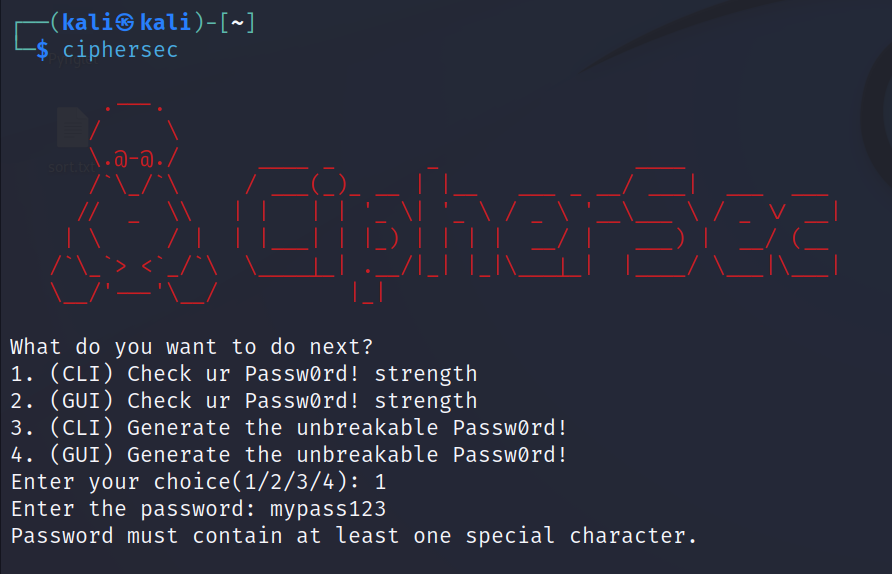


Figure 2.2: CLI interface for password strength checker, Password must contain at least one special character.

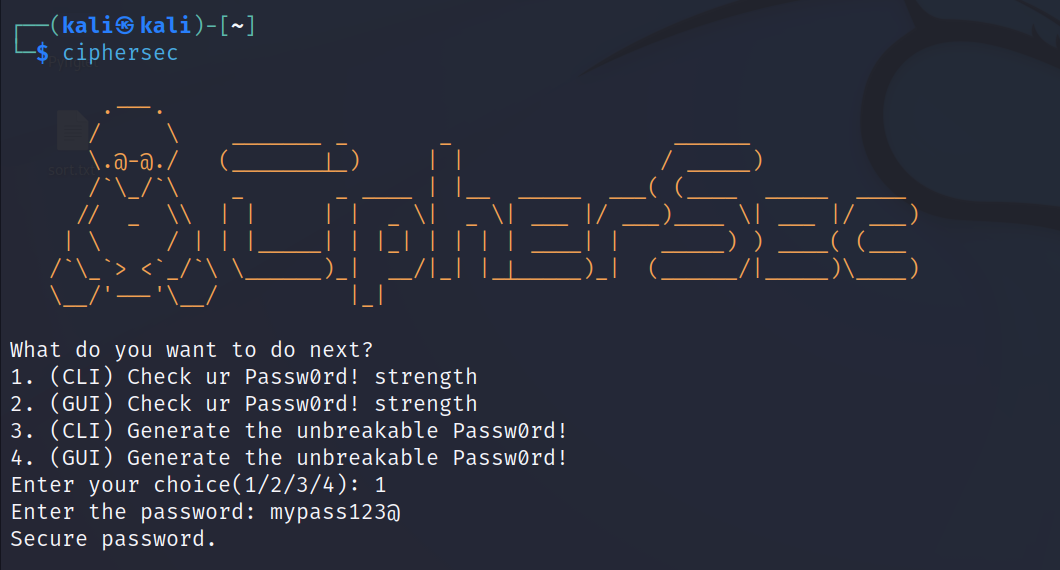


Figure 2.3: CLI interface for password strength checker, Secure password.

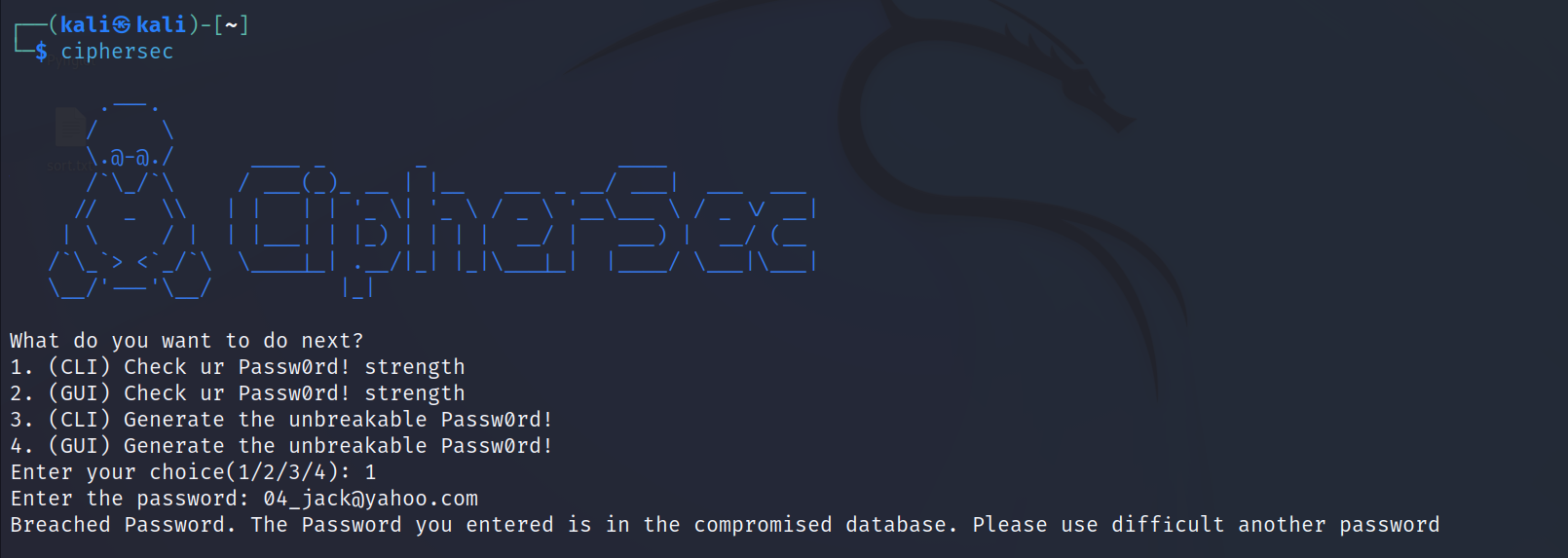


Figure 2.4: CLI interface for password strength checker, Breached Password. The Password you entered is in the compromised database. Please use difficult another password.

### **3. GUI interface for password strength checker:**

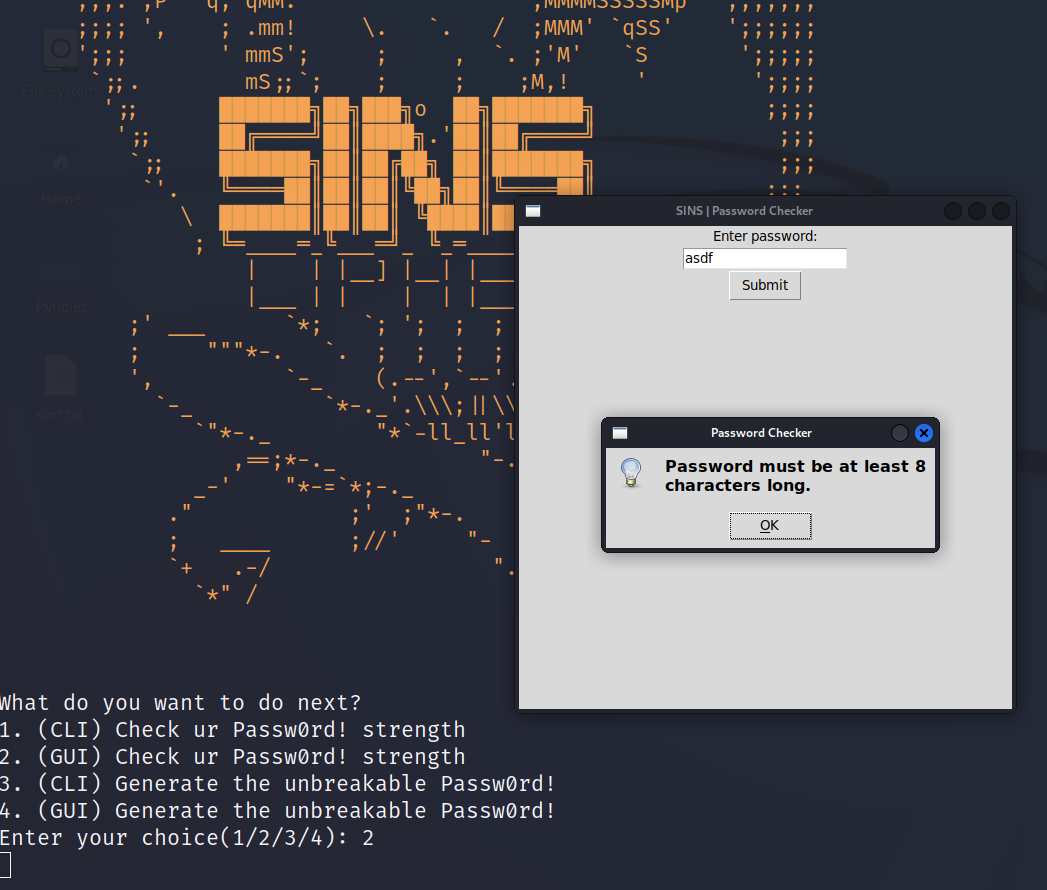


Figure : GUI interface for password strength checker.

Graphical user interface

Description automatically generated

Figure 3.1: GUI interface for password strength checker, Password must contain at least.

Graphical user interface

Description automatically generated

Figure 3.2: GUI interface for password strength checker, Password must contain at least one special character.

Graphical user interface

Description automatically generated

Figure 3.3: GUI interface for password strength checker, Secure password.

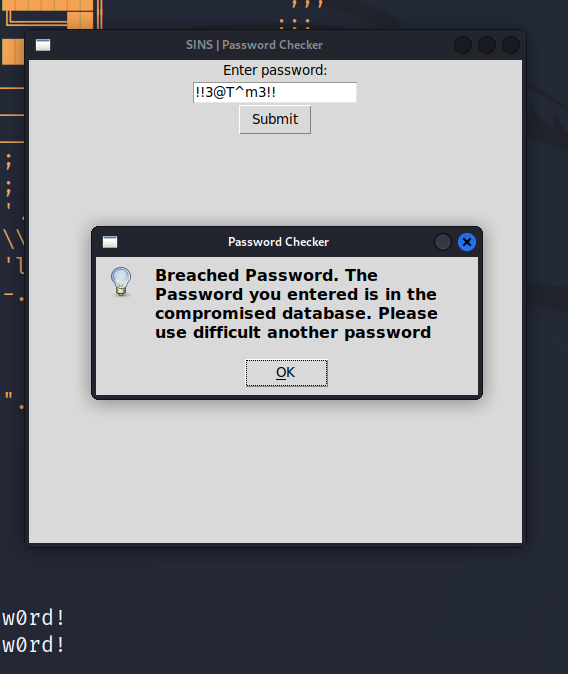


Figure 3.4: GUI interface for password strength checker, Breached Password. The Password you entered is in the compromised database. Please use difficult another password.

### **4. CLI interface for password generator:**



Figure 4.1: CLI interface for password generator.

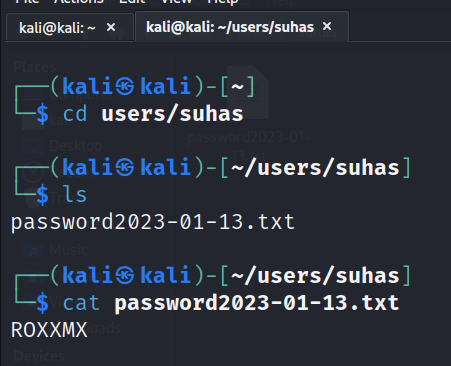


Figure 4.2: CLI interface for password generator, Password saved location.

Text

Description automatically generated

Figure .3: CLI interface for password generator, New Password.

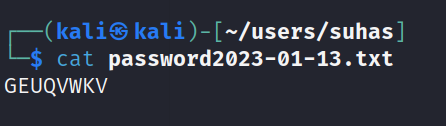


Figure 4.4: CLI interface for password generator, Saved Password in new file.



Figure 4.5: CLI interface for password generator, New user.

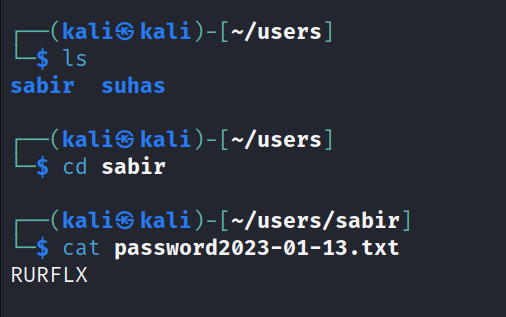


Figure 4.6: CLI interface for password generator, New user saved password location.

### **5. GUI interface for password generator:**



Figure .1: GUI interface for password generator, Password saved.

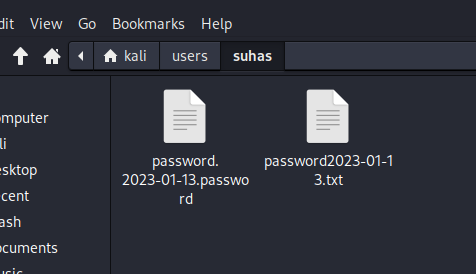


Figure 5.2: GUI interface for password generator, Password saved location.