

Projet 9: Maîtrise de python avancé

Damian Baerts, Baptiste Blet

March 8, 2024

1 Sommaire

- 1.1 Présentation du programme
- 1.2 Utilisation du programme
- 1.3 Présentation visuelle du programme
- 1.4 Explication du code
- 1.5 Améliorations
- 1.6 Disclaimer

2 Présentation du programme

Ce programme s'inscrit dans le cadre du projet scolaire "Maîtrise de python avancée" et est par conséquent développé en python. Il nous a été demandé de se fier à la database de "Have I been pwned?", ses résultats sont donc basés sur les réponses de cette API. Il possède 6 fonctionnalités différentes:

- "Password analyser" : détecteur de mot de passe compromis.
- "File analyser" : détecte si des mots de passes ont été compromis au sein d'un fichier texte.
- "Password generator" : générateur de mot de passe inconnu de la base de donnée "Have i been pwned".
- Une fonction qui détail les commandes relatives à son utilisation.
- Un générateur de bot Discord comprenant les fonctionnalités "File analyser", "Password analyser" et "Password generator".
- Une interface graphique pour les clients ne souhaitant pas se restreindre au terminal et faciliter l'importation de fichier textes.

3 Utilisation du programme

Pour son lancement en ligne de commande, si l'utilisateur entre le nom du programme sans arguments, le programme retourne sa syntaxe d'utilisation avec les arguments adéquats, à savoir :

- `python Pwned.py -p password` password représente le mot de passe à tester.
 - `python Pwned.py -f FileName.txt` FileName.txt représente le nom du fichier texte.
 - `python Pwned.py -g` Cette fonction vous demandera la longueur du mot de passe souhaitée pour le générer
 - `python Pwned.py -e` Retourne les explications du programme avec exemples.
 - `python Pwned.py -b` Lance le bot discord.
 - `python Pwned.py -wizard` Lance l'interface graphique depuis le terminal.
- L'exécution de ce programme nécessite l'installation de Python pour pouvoir le compiler et le lancer. il nécessite également l'installation des bibliothèques suivantes :
- requests
 - hashlib

- art
- termcolor
- os
- argparse
- random
- PIL
- subprocess

La commande pour les installer se fait dans le terminal :

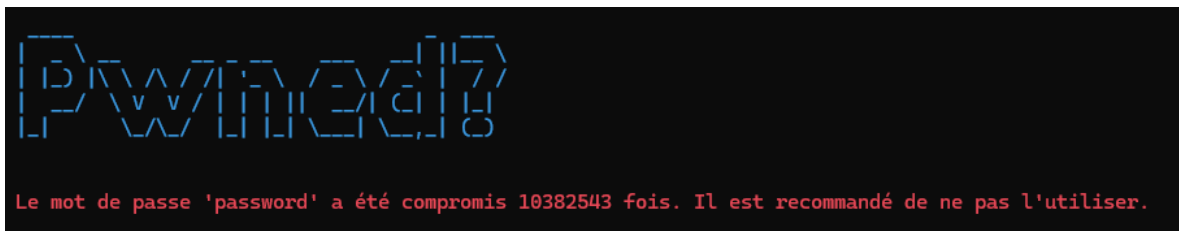
`pip install [bibliothèque 1][bibliothèque 2][bibliothèque 3] etc...`

Il est recommandé de vérifier leur présence avant l'exécution du programme.

4 Présentation visuelle du programme

4.1 Tester un seul mot de passe

La commande : `python PWNEd.py -p password` retourne l'affichage suivant :



```

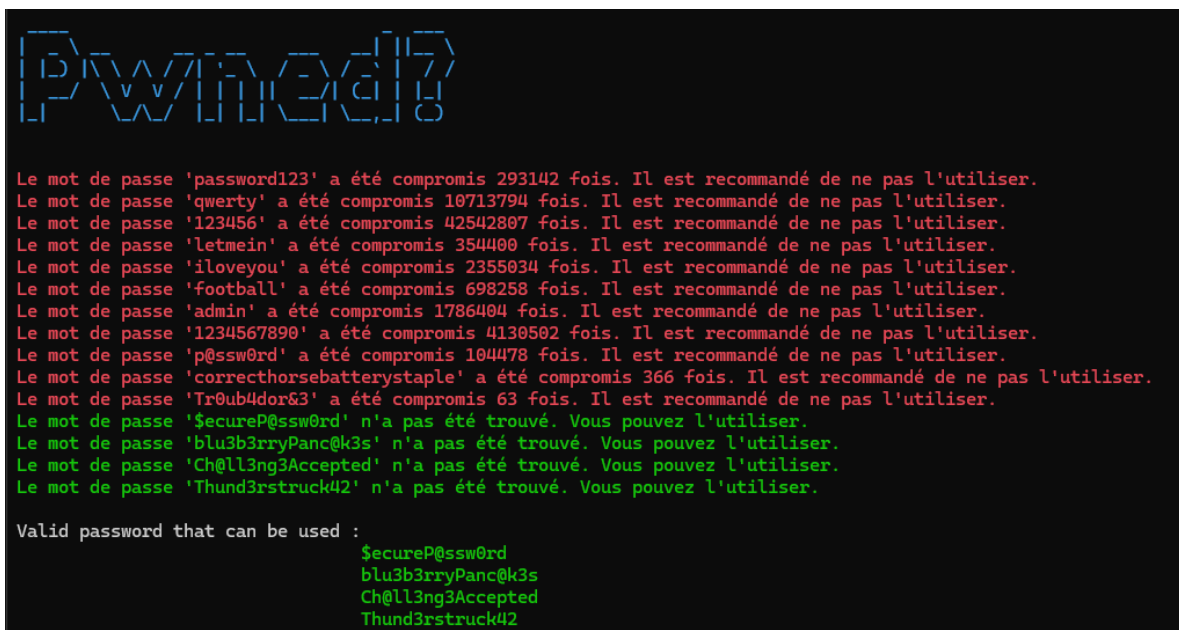
PWNEd

Le mot de passe 'password' a été compromis 10382543 fois. Il est recommandé de ne pas l'utiliser.
  
```

Avec la commande `HashIdProject.exe -f`, le programme va analyser à la suite toutes les lignes d'un fichier texte et les afficher à la suite en précisant leur index et en les réécrivant pour être sûr de leur équivalent.

4.2 Tester plusieurs mot de passe dans un fichier texte

L'utilisation de la commande : `python PWNEd.py -f FileName.txt` retourne l'affichage suivant :



```

PWNEd

Le mot de passe 'password123' a été compromis 293142 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'qwerty' a été compromis 10713794 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe '123456' a été compromis 42542807 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'letmein' a été compromis 354400 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'iloveyou' a été compromis 2355034 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'football' a été compromis 698258 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'admin' a été compromis 1786404 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe '1234567890' a été compromis 4130502 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'p@ssw0rd' a été compromis 104478 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'correcthorsebatterystaple' a été compromis 366 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe 'Tr0ub4dor&3' a été compromis 63 fois. Il est recommandé de ne pas l'utiliser.
Le mot de passe '$ecureP@ssw0rd' n'a pas été trouvé. Vous pouvez l'utiliser.
Le mot de passe 'blu3b3rryPanc@k3s' n'a pas été trouvé. Vous pouvez l'utiliser.
Le mot de passe 'Ch@ll3ng3Accepted' n'a pas été trouvé. Vous pouvez l'utiliser.
Le mot de passe 'Thund3rstruck42' n'a pas été trouvé. Vous pouvez l'utiliser.

Valid password that can be used :
    $ecureP@ssw0rd
    blu3b3rryPanc@k3s
    Ch@ll3ng3Accepted
    Thund3rstruck42
  
```

Il est important que dans le fichier texte, les mots de passe soient écrits ligne par ligne, des caractères pour les séparer pourraient être considérés comme la continuité du mot de passe. Cette fonction ne retourne pas que les mots de passe valides pour que l'utilisateur puisse se rendre compte du nombre de fois où ceux qui sont retournés en rouge ont été compromis. Les mots de passe valides sont cependant retournés en vert à la fin pour faciliter leur repérage et les copier coller si nécessaire.

4.3 Générer un mot de passe non-compromis

L'utilisation de la commande : `python Pwned.py -g` provoquera l'affichage de :



```
Pwned

Quelle est la longueur du mot de passe que vous voulez générer : |
```

Il faudra que l'utilisateur entre le nombre de caractères souhaités dans son mot de passe pour que le générateur lui en retourne un qui n'a jamais été compromis comme ci-dessous.



```
Pwned

Quelle est la longueur du mot de passe que vous voulez générer : 15
Le mot de passe généré est : ^X5zZ~N#7Gw@)-e
Le mot de passe n'a pas été trouvé. Vous pouvez l'utiliser.
```

4.4 Explication du programme

L'utilisation de la commande : `python PWNED.py -e / -h` (ou bien juste : `python PWNED.py`) provoque l'affichage de la syntaxe d'utilisation :

```
PWNED?

Guide d'utilisation :

Syntaxe : python3 .\fichier_de_test.py [argument]

liste d'arguments : -p : permet de passer directement un mot de passe au programme
                    Exemple : python3 .\fichier_de_test.py -p 'mdp'

                    -f : permet de passer un fichier au programme en précisant son chemin d'accès afin de traiter
                        plusieurs mots de passe
                    Exemple : python3 .\fichier_de_test.py -f 'path_to_file'

                    -g : permet de générer un mot de passe et de vérifier qu'il n'a pas fuité en le passant au programme
                    Exemple : python3 .\fichier_de_test.py -g

                    -b : permet de lancer le bot discord
                    Exemple : python3 .\fichier_de_test.py -b

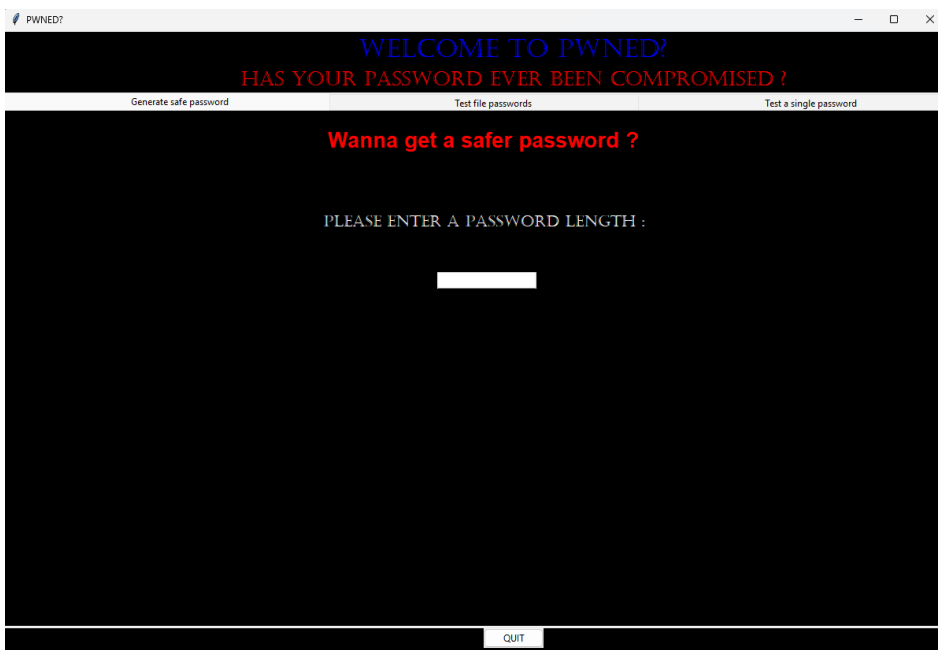
                    -wizard : permet de lancer le programme en mode graphique
                    Exemple : python3 .\fichier_de_test.py -wizard
```

Si l'utilisateur possède quelques connaissances des programmes informatiques homemade, il sera tenté de rentrer -h pour en apprendre plus, si en revanche il n'en a pas l'habitude, il entrera juste le nom du programme et verra s'afficher la syntaxe à utiliser avec exemples. Il est important de ne pas négliger les nouveaux utilisateurs.

4.5 Interface Graphique

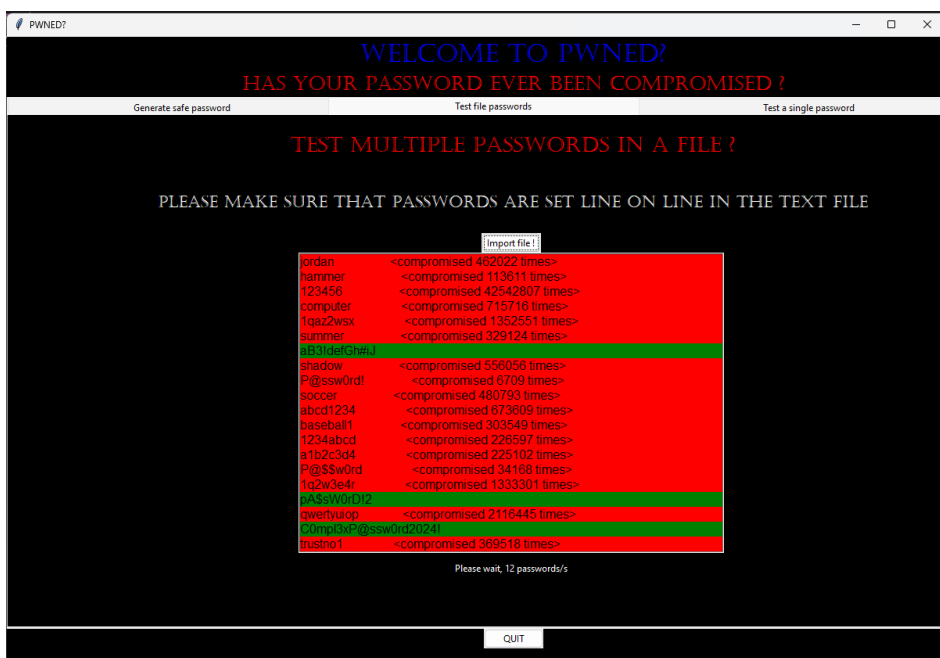
L'utilisation de la commande : `python Pwned.py -wizard` provoque l'affichage d'une interface graphique pour les utilisateurs qui ne sont pas très à l'aise avec les lignes de commande où qui souhaitent faciliter l'importation de fichier.

- 1. "Password generator"



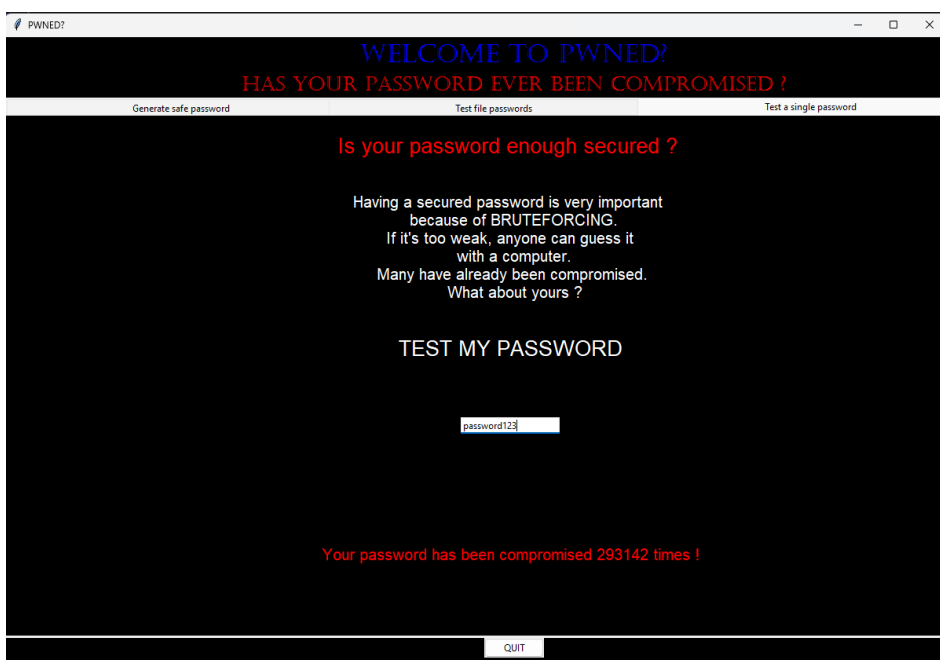
Cette première page de l'interface est dédiée au "Password generator" qui va vous renvoyer un mot de passe non compromis après que vous ayez choisi sa longueur souhaitée.

- 2. "File analyser"



Cette page est dédiée au 'File analyser" et va analyser tout un fichier texte pour en analyser les mots de passe. Il est précisé que cette méthode analyse 12 mots de passe par secondes. Il n'est possible d'importer que des fichiers textes pour éviter tout problème.

- 3. "Password analyser"



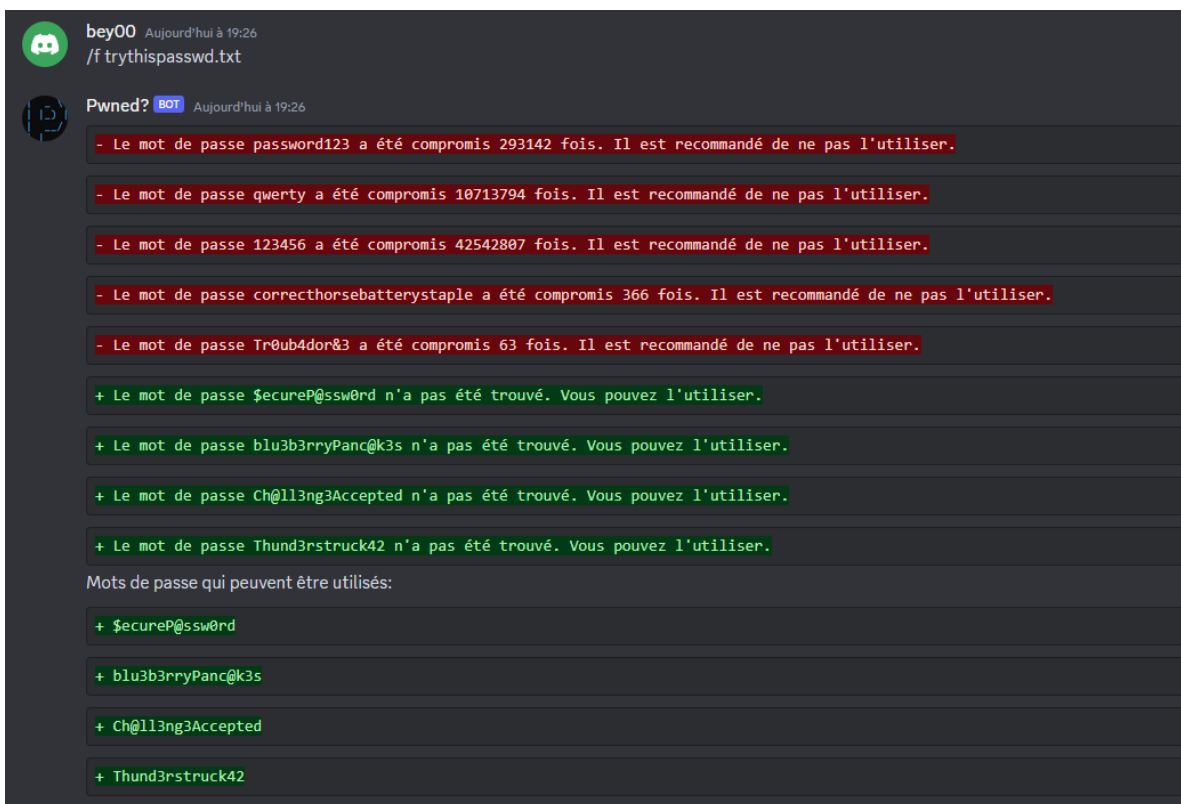
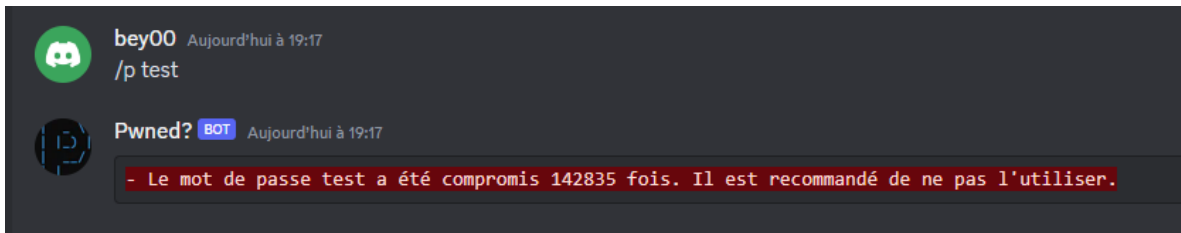
Cette page est dédiée au test d'un seul mot de passe. Dans le cas où il n'est pas répertorié, la page retournera la mention "Greetings !" écrite en vert, dans le cas contraire, elle retournera le nombre de fois où il a été compromis cette fois-ci, en rouge.

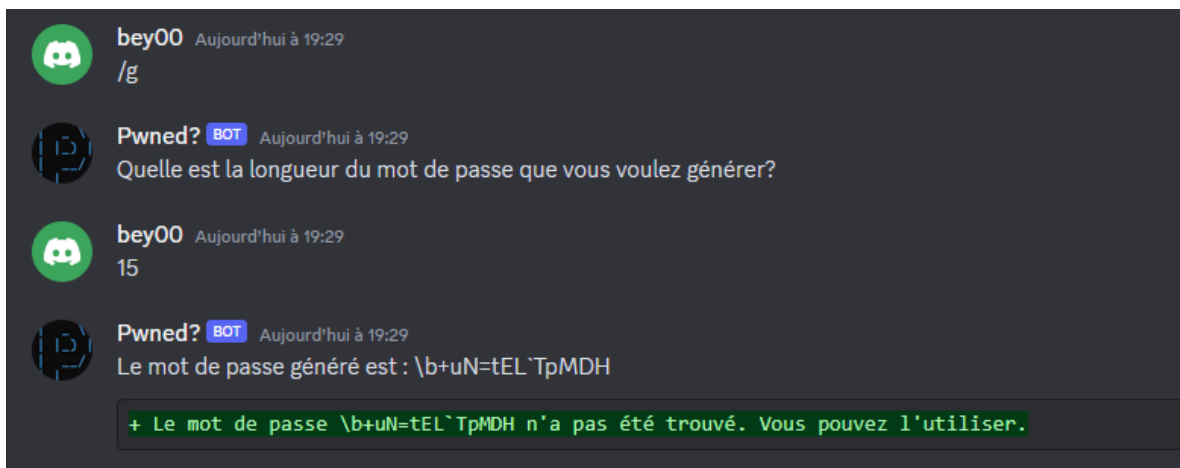
4.6 BotDiscord

Pour cette étape les prérequis seront détaillés en page 10. Avec la commande : `python Pwned.py -b`, le programme vous demandera le token du bot discord auquel vous souhaitez attribuer les fonctionnalités décrites plus haut.

```
Vous devez avoir créé un bot Discord pour cette fonctionnalité
Entrez exit si ce n'est pas fait.
Veuillez entrer le token du bot discord : MTIxNDkyOTE4ODA3MTY3ODAxNA.GIS5rd.bqDe1xj34AdodimSibc64ch_g38u1Bfv9A_ga4
[2024-03-08 19:14:47] [INFO] discord.client: logging in using static token
[2024-03-08 19:14:49] [INFO] discord.gateway: Shard ID None has connected to Gateway (Session ID: 6f6db85c674d2a421fb8080492a23646).
Bot is ready
```

Une fois cette étape faite, le bot est prêt pour son utilisation.
Voici les commandes qui peuvent être effectuées.





/p "MdpATester" pour analyser un unique mot de passe.

/f "PathToFile/FileName.txt"

/g puis entrez une valeur pour générer un mot de passe de la longueur souhaitée

5 Explication du code

5.1 Les bibliothèques nécessaires

Voici les bibliothèques nécessaires au bon fonctionnement du programme. Leur méthode d'installation est décrite page 1.

```
import requests
import hashlib
from art import text2art
from termcolor import colored
import os
import argparse
import random
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import filedialog
import subprocess
```

5.2 Clear console

```
def clearConsole():
    return os.system("cls" if os.name in ("nt", "dos") else "clear")

clearConsole()
```

Cette fonction permet de clear le terminal afin de rendre l'affichage du programme plus agréable.

5.3 Check password

```
def check_password(password):
    sha_password = hashlib.sha1(password.encode()).hexdigest()
    sha_prefix = sha_password[:5]
    sha_postfix = sha_password[5:].upper()

    url = "https://api.pwnedpasswords.com/range/" + sha_prefix

    try:
        response = requests.get(url)

        pwnd_dict = {}
        pwnd_list = response.text.split("\r\n")
        for pwnd_pass in pwnd_list:
            pwnd_hash = pwnd_pass.split(":")
            pwnd_dict[pwnd_hash[0]] = pwnd_hash[1]

        if sha_postfix in pwnd_dict.keys():
            return int(pwnd_dict[sha_postfix])
        else:
            return 0
    except requests.exceptions.RequestException as e:
        print("\033[91mErreur de connexion au service Pwned Passwords :", e, "\033[0m")
        return -1
```

Cette fonction représente le point d'orgue du programme, elle envoie les requêtes à l'API "have i been pwned?" et en ressort le nombre de fois où le mot de passe entré a été compromis. Toutes les autres commandes se basent sur les résultats de cette fonction.

5.4 password test

```
def MyPassword(ThePassword):
    print(colored(text2art("Pwned?"), "cyan"))
    compromised_count = check_password(ThePassword)
    if compromised_count > 0:
        print("\033[91mLe mot de passe '{}' a été compromis {} fois. Il est recommandé de ne pas l'utiliser.\033[0m".format(args.password, compromised_count))
    else:
        print("\033[92mLe mot de passe n'a pas été trouvé. Vous pouvez l'utiliser.\033[0m")
```

Cette fonction teste un mot de passe en argument et renvoie le nombre de fois où il a été compromis.

5.5 File test

```
def CheckFile(TheFilePath):
    try:
        print(colored(text2art("Pwned?"), "cyan"))
        ValidPasswd = []
        with open(TheFilePath, 'r') as f:
            passwords = f.readlines()
        for password in passwords:
            password = password.strip()
            compromised_count = check_password(password)
            if compromised_count > 0:
                print("\033[91mLe mot de passe '{}' a été compromis {} fois. Il est recommandé de ne pas l'utiliser.\033[0m".format(password, compromised_count))
            else:
                print("\033[92mLe mot de passe '{}' n'a pas été trouvé. Vous pouvez l'utiliser.\033[0m".format(password))
                ValidPasswd.append(password)
        print("\nValid password that can be used :",end='\n')
        for PasswordValid in ValidPasswd:
            print("\033[92m                {}\033[0m".format(PasswordValid))
    except FileNotFoundError:
        print("\033[91mLe fichier spécifié n'a pas été trouvé.\033[0m")
```

Cette fonction va analyser un fichier texte et utiliser la fonction checkpassword sur chaque ligne qui est censée représenté un mot de passe comme décrit plus haut, puis va en retourner le nombre de fois ou ils ont été compromis pour chacun.

5.6 Generate password

```
def GenPassword():
    print(colored(text2art("Pwned?"), "cyan"))
    mdp_range = int(input("Quelle est la longueur du mot de passe que vous voulez générer : "))
    if mdp_range <= 0:
        print("\033[91mLa longueur du mot de passe doit etre superieur a 0.\033[0m")
        mdp_range = int(input("Quelle est la longueur du mot de passe que vous voulez générer : "))
    else:
        while True:
            mdp_generate = ""
            for i in range(mdp_range):
                character = chr(random.randint(32, 126))
                mdp_generate += character
            print(f"Le mot de passe généré est : {mdp_generate}")

            compromised_count = check_password(mdp_generate)
            if compromised_count > 0:
                print("\033[91mLe mot de passe '{}' a été compromis {} fois. Il est recommandé de ne pas l'utiliser.\033[0m".format(mdp_generate, compromised_count))
            else:
                print("\033[92mLe mot de passe n'a pas été trouvé. Vous pouvez l'utiliser.\033[0m")
                break
```

Cette fonction va demander une longueur de mot de passe afin de pouvoir en générer un grâce au module random, puis le tester dans l'API Have i been pwned? Si il n'a jamais été compromis, le retourne dans le terminal.

5.7 Explication Argument

```
def Explain():
    print(colored(text2art("Pwned?"), "cyan"))
    print("Guide d'utilisation :\n")
    print("Syntaxe : python3 .\\Pwned.py [argument]\n")
    print("liste d'arguments : -p : permet de passer directement un mot de passe au programme")
    print("                    \033[92mExemple : python3 .\\fichier_de_test.py -p 'mdp'\n\033[0m")
    print("                    -f : permet de passer un fichier au programme en précisant son chemin d'accès afin de traiter\n                                \033[92mExemple : python3 .\\fichier_de_test.py -f 'path_to_file'\n\033[0m")
    print("                    -g : permet de générer un mot de passe et de vérifier qu'il n'a pas fuité en le passant au programme")
    print("                    \033[92mExemple : python3 .\\fichier_de_test.py -g\n\033[0m")
    print("                    -b : permet de lancer le bot discord")
    print("                    \033[92mExemple : python3 .\\fichier_de_test.py -b\n\033[0m")
    print("                    -wizard : permet de lancer le programme en mode graphique")
    print("                    \033[92mExemple : python3 .\\fichier_de_test.py -wizard\n\033[0m")
```

Cette fonction provoque l'affichage de la syntaxe à utiliser pour le fonctionnement du programme, elle est nécessaire si une personne ne possède pas la documentation et son affichage se fait même sans argument.

5.8 GraphicUser

```
class GraphUser:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("PMMED?")
        self.root.configure(bg='#000000')
        self.root.geometry("1200x800")
        self.cool = ttk.Frame(self.root)

        self.notebook = ttk.Notebook(self.cool)

        style = ttk.Style()
        style.configure("Custom.TFrame", background='black')

        self.tab1 = ttk.Frame(self.notebook, height = 660, style = "Custom.TFrame")
        self.tab2 = ttk.Frame(self.notebook, height = 660, style = "Custom.TFrame")
        self.tab3 = ttk.Frame(self.notebook, height = 660, style = "Custom.TFrame")
        self.notebook.add(self.tab1, text="Generate safe password")
        self.notebook.add(self.tab2, text="Test file passwords")
        self.notebook.add(self.tab3, text="Test a single password")
        self.ExitButton = ttk.Button(self.root, command = self.root.destroy, text = 'QUIT').grid(row = 2)
        self.notebook.grid()
        self.cool.grid(row = 1)
        self.entry = ttk.Entry(self.tab1)
        self.ValeurEntree = ttk.Label(self.tab1, text="Please enter a password length : ", background = "black", foreground = "white", font=("Castellar", 15))
        self.CheckMyPwd = ttk.Label(self.tab3, text = "", foreground = "white", background = "black")
        self.SecondPageFrame = ttk.Frame(self.tab2, width = 500, height = 300, style = 'Custom.TFrame', padding = "5")
        self.ResultFrame = tk.Listbox(self.SecondPageFrame, height = 20, width = 60, background = 'black', foreground = 'black', font = ('Helvetica', 12))
        self.ErrorInOpeningFile = ttk.Label(self.SecondPageFrame, text = "")
        self.Bienvenue()
        self.PremierePage()
        self.SecondPage()
        self.ThirdPage()
```

Cette fonction comprend toute l'interface graphique, elle n'est que partiellement montrée dans la documentation. Lire le code pour les détails.

5.9 Bot Discord

```
bot = commands.Bot(command_prefix='/', intents=discord.Intents.all())

@bot.event
async def on_ready():
    print("Bot is ready")

@bot.command()
async def p(ctx: commands.Context, *, message: str):
    compromised_count = check_password(message)
    if compromised_count > 0:
        await ctx.send("```diff\n- " + "Le mot de passe " + message + " a été compromis " + str(compromised_count) + " fois. Il est recommandé de ne pas l'utiliser." + "\n```")
    else:
        await ctx.send("```diff\n+ " + "Le mot de passe " + message + " n'a pas été trouvé. Vous pouvez l'utiliser." + "\n```")

@bot.command()
async def f(ctx: commands.Context, file: str):
    try:
        ValidPasswd = []
        with open(file, 'r') as f:
            passwords = f.readlines()
            for password in passwords:
                password = password.strip()
                compromised_count = check_password(password)
                if compromised_count > 0:
                    await ctx.send("```diff\n- " + "Le mot de passe " + password + " a été compromis " + str(compromised_count) + " fois. Il est recommandé de ne pas l'utiliser." + "\n```")
                else:
                    await ctx.send("```diff\n+ " + "Le mot de passe " + password + " n'a pas été trouvé. Vous pouvez l'utiliser." + "\n```")
                    ValidPasswd.append(password)
        await ctx.send("\nMots de passe qui peuvent être utilisés:")
        for PasswordValid in ValidPasswd:
            await ctx.send("```diff\n+ PasswordValid + "\n```")
    except FileNotFoundError:
        print("\033[91mLe fichier spécifié n'a pas été trouvé.\033[0m")

@bot.command()
async def g(ctx: commands.Context):
    await ctx.send("Quelle est la longueur du mot de passe que vous voulez générer?")
    def check(message):
        return message.author == ctx.author and message.channel == ctx.channel and message.content.isdigit()
    try:
        msg = await bot.wait_for('message', timeout=60.0, check=check)
        mdp_range = int(msg.content)
    except asyncio.TimeoutError:
        await ctx.send("Temps écoulé. Veuillez réessayer.")
        return
    except ValueError:
        await ctx.send("Veuillez saisir un nombre valide.")
        return
    if mdp_range <= 0:
        await ctx.send("La longueur du mot de passe doit être supérieure à 0.")
    else:
        while True:
            mdp_generate = ""
            for i in range(mdp_range):
                character = chr(random.randint(32, 126))
                mdp_generate += character
            await ctx.send("Le mot de passe généré est : " + mdp_generate)

            compromised_count = check_password(mdp_generate)
            if compromised_count > 0:
                await ctx.send("```diff\n- " + "Le mot de passe " + mdp_generate + " a été compromis " + str(compromised_count) + " fois. Il est recommandé de ne pas l'utiliser." + "\n```")
            else:
                await ctx.send("```diff\n+ " + "Le mot de passe " + mdp_generate + " n'a pas été trouvé. Vous pouvez l'utiliser." + "\n```")
                break

if __name__ == "__main__":
    while True:
        print("Vous devez avoir créé un bot Discord pour cette fonctionnalité.\nEnterrez exit si ce n'est pas fait.")
        BotToken = input("Veuillez entrer le token du bot discord : ")
        if BotToken == "exit":
            break
        elif BotToken:
            try:
                bot.run(BotToken)
            except:
                print("WRONG TOKEN.")
                break
```

Tout ce code se trouve dans un fichier à part et sert à l'initialisation du bot, ses fonctions de renvoi dans le chat, et ses analyses de mot de passes ainsi que de fichiers et de génération de mot de passe non compromis. Toujours en se basant sur la fonction CheckPassword. Pour le bon fonctionnement, doit se trouver dans le même dossier que Pwned.Py.

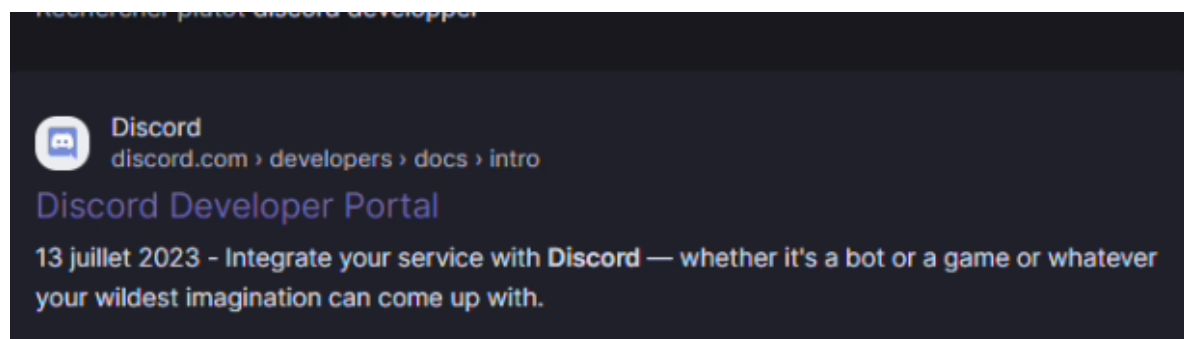
5.10 Fonction main

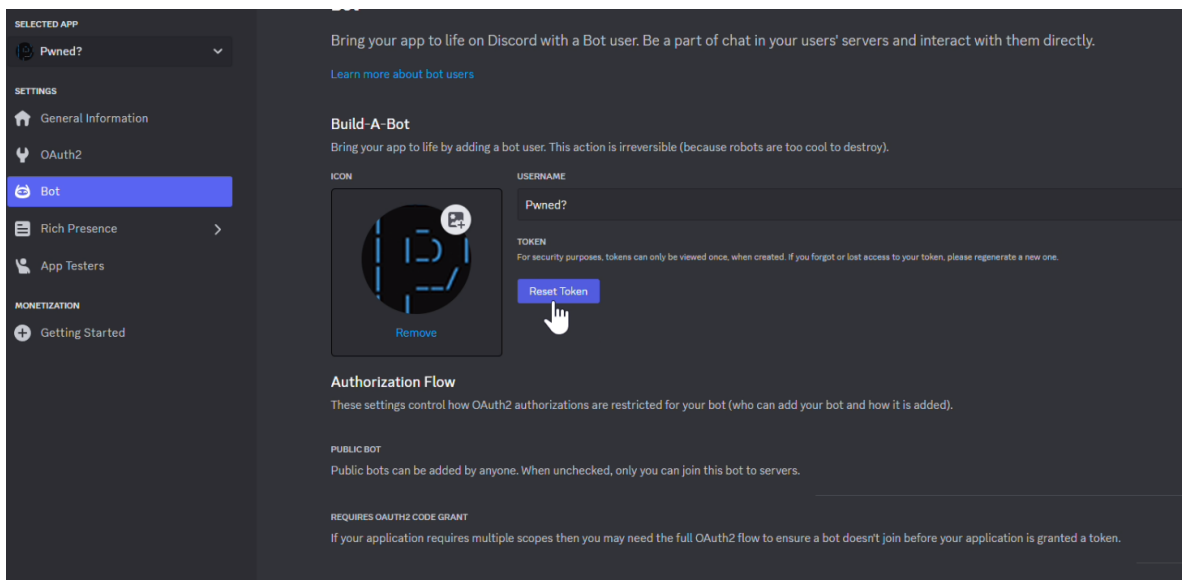
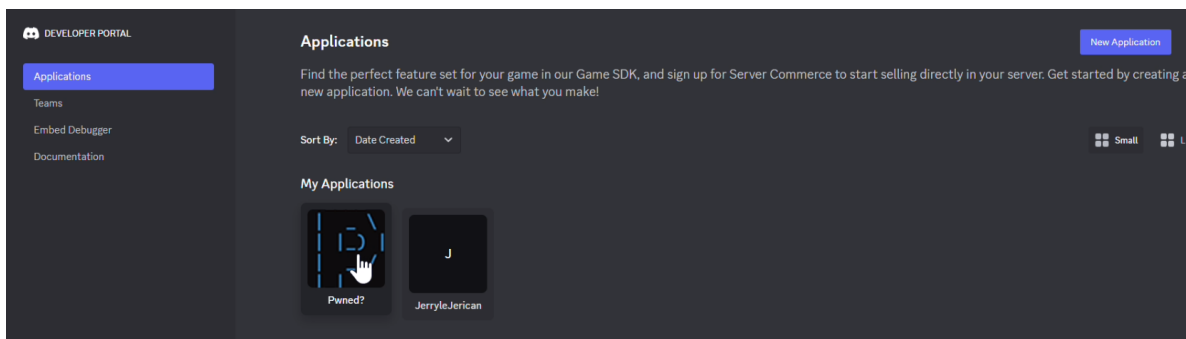
```
def main():
    if args.password:
        MyPassword(args.password)
    elif args.file:
        CheckFile(args.file)
    elif args.generate:
        GenPassword()
    elif args.explication:
        Explain()
    elif args.wizard:
        GraphUser().start()
    elif args.botdiscord:
        script_path = 'bot.py'
        subprocess.run(['python', script_path])
    else:
        Explain()

if __name__ == "__main__":
    main()
```

Pour finir, la fonction main, chargée d'assurer le lancement des fonctions adéquates à l'argument entré par l'utilisateur, dans le cas où il n'y a pas d'argument ou bien un argument non-existant, renvoie la fonction Explain.

6 Discord bot token





7 Améliorations

Ce programme a été réalisé en 12 jours et n'est par conséquent pas totalement complet. Rien ne l'est jamais.

Voici quelques améliorations qui pourraient être intéressantes :

- Amélioration du système de requêtes
- Amélioration de l'interface graphique
- Implémentation de nouvelles fonctionnalités

8 Disclaimer

Ce programme s'inscrit dans le cadre d'une future présentation devant jury mais a été réalisée par deux étudiants en première année, aussi sa complexité reste modérée et n'est en aucun cas plus précise ou efficace que d'autres équivalents officiels.

Merci à l'API "Have I been Pwned?" qui a assurée par sa base de donnée le bon fonctionnement de notre programme.

Réalisé dans le cadre du projet "Maîtrise de python avancée".

Enjoy programming is the way to make better projects. Again.

