REPORT

- Navodita Mathur

Objective:

The goal of this programming project is to implement a password authentication mechanism and a password cracker to study the vulnerabilities of choosing weak passwords.

Description:

The project has a mechanism that registers and adds a new user into the system and stores the user's password information in a file. The password can contain only lower-case letters and a maximum of 2 numbers and 2 special characters ('@', '#', '$', '%', '&'). Security is ensured by storing not the password strings but message digests (hash) of the passwords to prevent attacks. It also allows registered users to login using a module which asks for the username and password from the user and verifies it based on the information stored in the password file. It computes the MD5 message digest of the entered password and checks if it matches the MD5 digest of the corresponding user password stored in the password file. The program accepts the user only if the message digests match.

The project also offers mechanisms to crack the password of any existing user with a known username based on a dictionary of commonly used words. Users logged in can also validate their passwords and know the time taken to crack their password and how strong it is.

Deployment:

http://nam266.pythonanywhere.com/

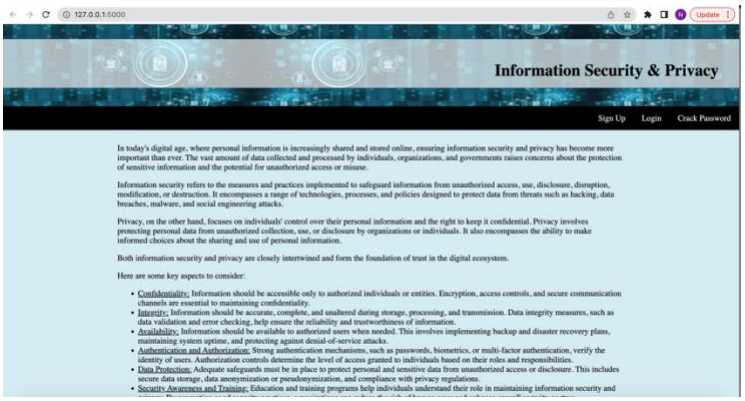
Code Repository:

https://github.com/Navoditamathur/password_cracker

Implementation:

The project is a website having mechanisms bar to register and login and crack passwords of registered users.

PART-1:



- User Registration:

Users are required to give details like their first name, last name, email-id, a unique username, and password.
For example:

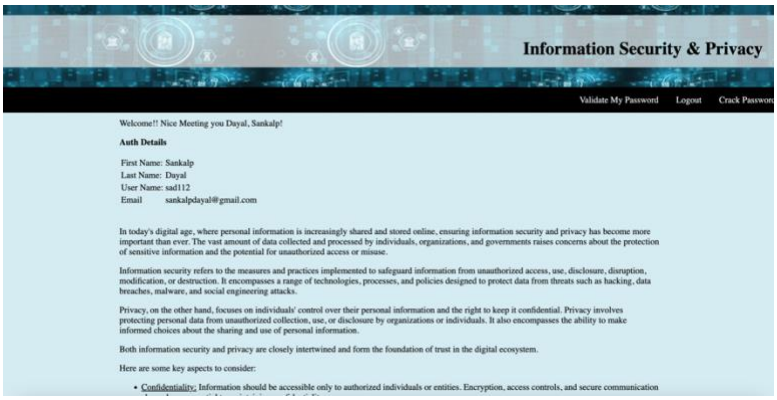
First Name	Sankalp
Last Name	Dayal
Username	sad112
E-mail	Sankalpdayal@gmail.com
Password	acacia

It saves the details to text files.

- User Login

After successful registration, the user is redirected to login screen.

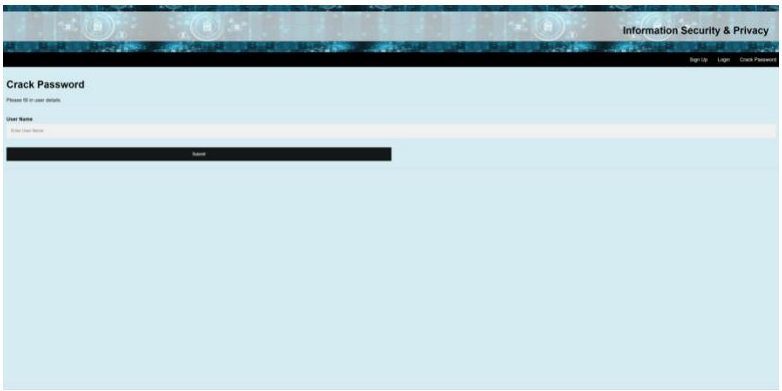
If the login is successful, the user details are displayed on the screen.



Otherwise, appropriate error message is shown.

PART-2:

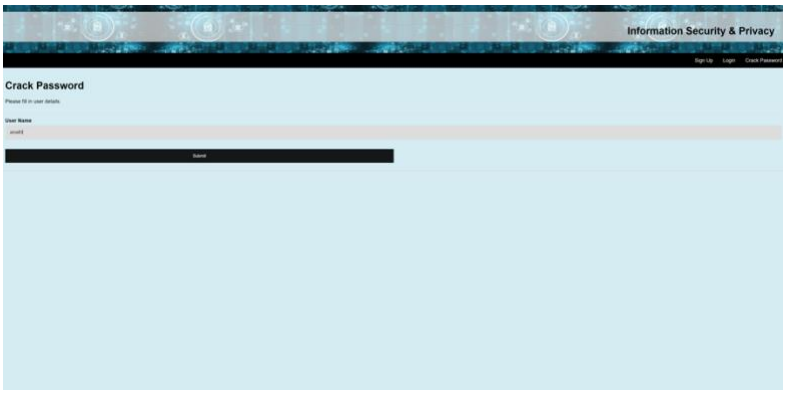
For this part, the user takes username as input from the user. Matching with the words present in the dictionary and then permutations and combinations of the word with numbers and special characters, up to 4 such characters. It displays the username, password and time taken to crack the password.

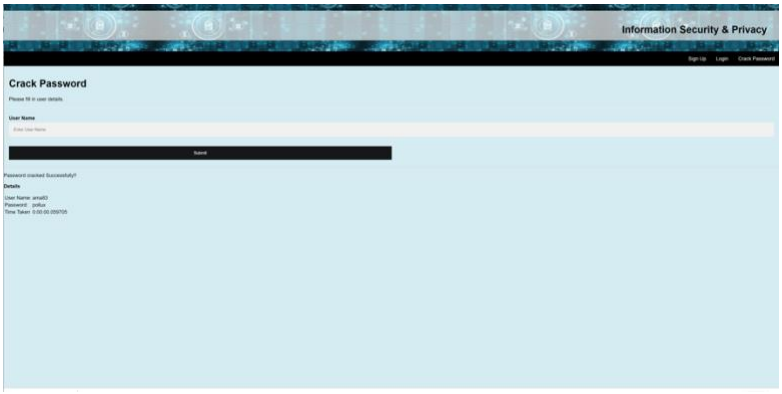


Type-1:

If the password string is just exactly one of the words present in the dictionary, it is of type 1. To avoid scenario where password starting with ‘z’ takes longer to crack than the password starting with ‘a’, the dictionary words from text file are taken into dictionary object and shuffled randomly.

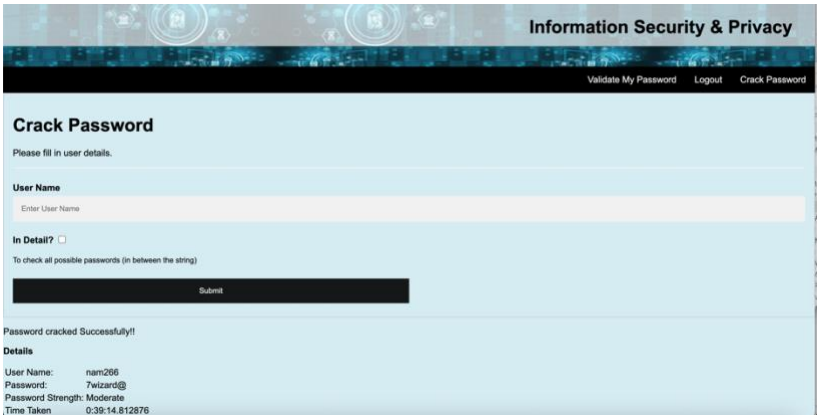
For example, user with username “ama83” has “pollux” as password and can be cracked within 2 seconds.





Type-2:
If the password string is a combination of a dictionary word, numerical characters 0-9 and special characters, {@, #, $, %, &}, then characters are iterated taking 4 maximum at a time, hashed and matched with hashed passwords stored in the text file.

For example, user with username “nam266” has “7wizard@” as password and can be cracked in 40 minutes.

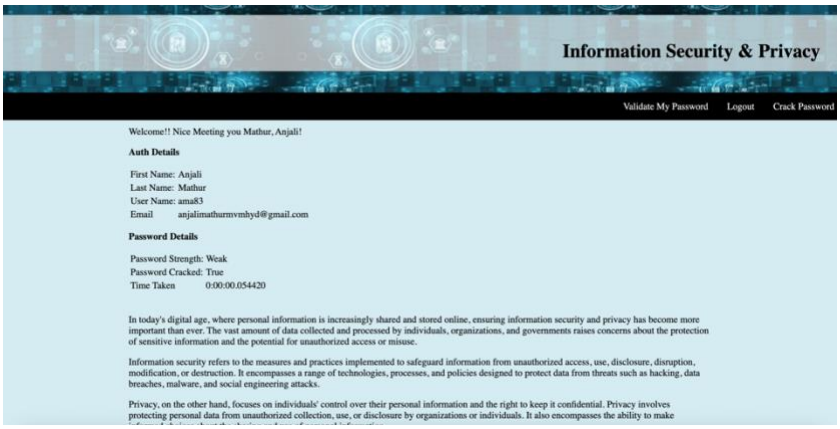


If the system is unable to crack the message, it displays a message saying that password cannot be cracked.

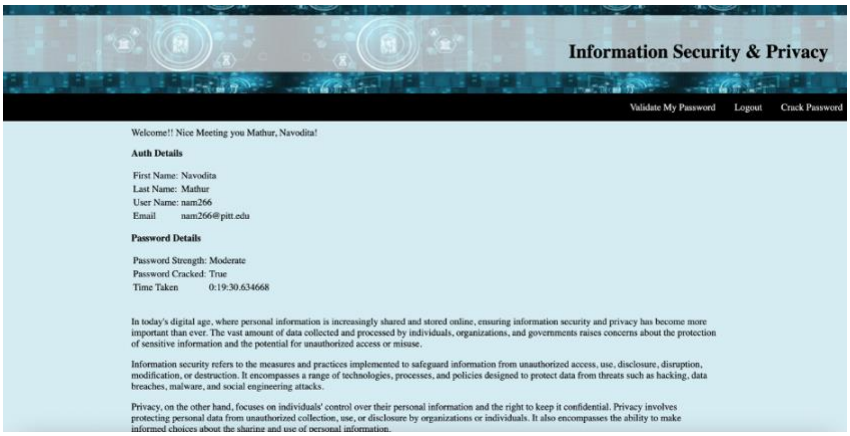
PART-3:

This can be done only by the logged-in users. Like Part-2, the system cracks the password by taking username stored in session. It displays Password strength, if cracked and time taken to crack the password (if cracked at all)

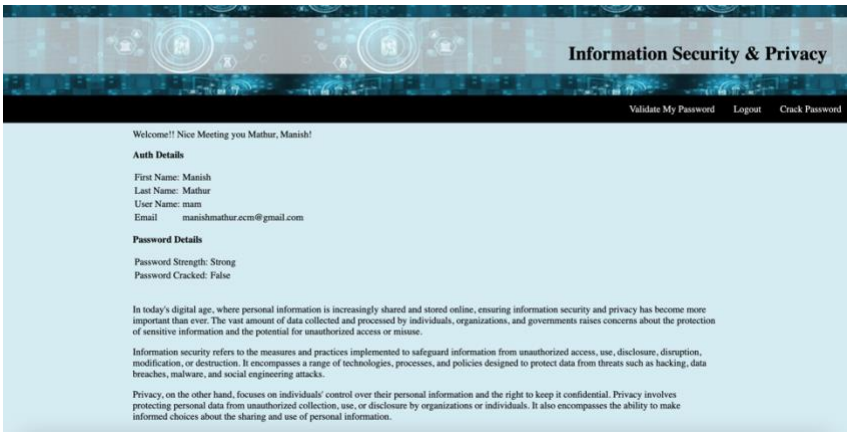
- Weak Passwords: Any password string that exactly matches a dictionary word is classified as a weak password. Example – User with username “ama83” has “pollux” as password.



- Moderate Password: Any password string does not exactly match a dictionary word but contains a dictionary word as a substring of the password string. Example – User with username “nam266” has “7wizard@” as password. It takes much longer and processing power to crack than weak password.



- Strong Password: A strong password does not contain any dictionary words as part of its substring. It cannot be cracked by the system using this method. Example – User with username “mam” has abcdef12@#” as password.



Conclusion

The more uncommon a word is chosen as password, the harder it is to crack. Increase in complexity of passwords with the help of numbers and special characters, leads to increase in time and processing power required to crack the password.