Integrated Security Vault

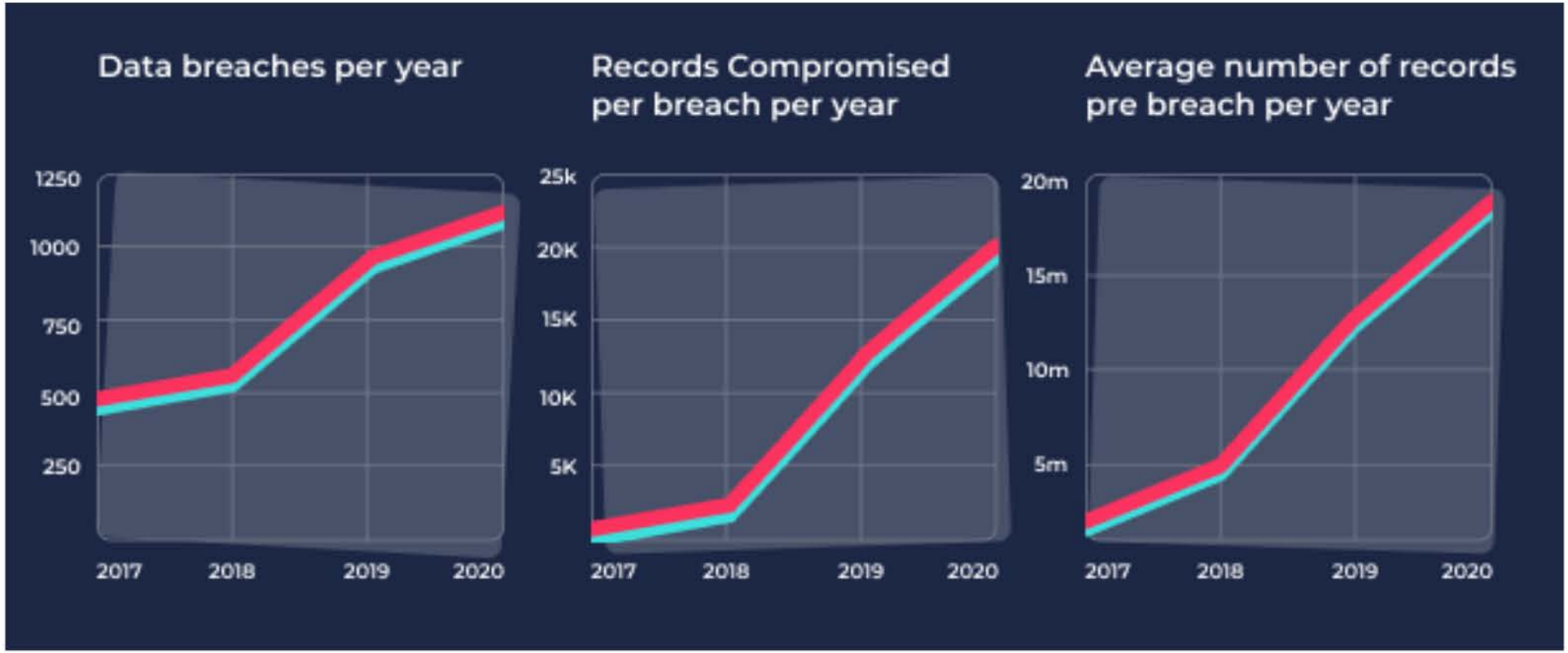
Hussain Harianawala

16010421130

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

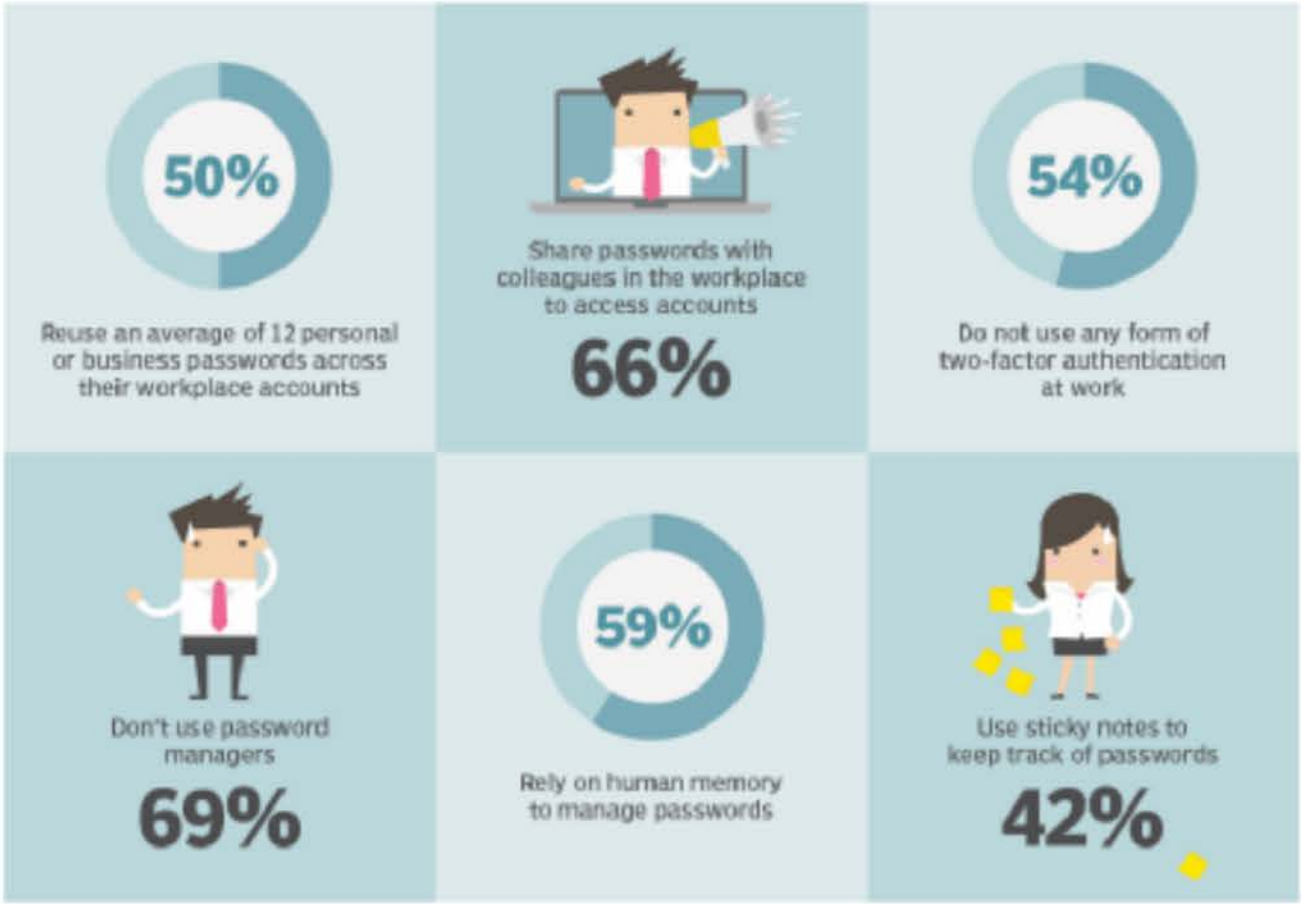
Somaiya TRUST

Why do we need this ?



Password hygiene shortcomings

The Ponemon Institute's latest research report on password practices reveals several areas where real-world practices fall short of password best practices.



Entropy Of a Password

- Password Entropy is the measure of password strength or how strong the given password is.
- It is a measure of effectiveness of a password against guessing or brute-force attacks.
- It decides whether the entered password is common and easily crack-able or not.
- It is calculated by knowing character set (lower alphabets, upper alphabets, numbers, symbols, etc.) used and the length of the created password. It is expressed in terms of bits of entropy per character.

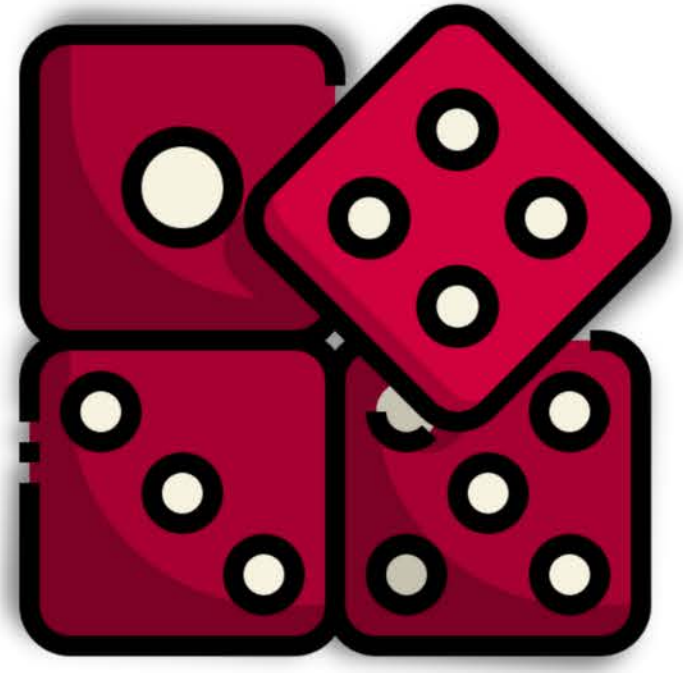
Entropy of a password

```python
from math import log2
import string

def entropy(password):

    R = 0
    L = len(password)
    if any(i in string.digits for i in password):
        R += 10
    if any(i in string.ascii_lowercase for i in password):
        R += 26
    if any(i in string.ascii_uppercase for i in password):
        R += 26
    if any(i in string.punctuation for i in password):
        R += 32
    # print(R)
    E = L * log2(R)
    G = int(2 ** ((E) - 1))  # average entropy
    T = int(G / (10 ** 11))
    print("\nEntropy : ", E)
```

Diceware "Passphrase"

- People often pick some phrase from pop culture — favorite lyrics from a song or a favorite line from a movie or book — and slightly mangle it by changing some capitalization or adding some punctuation or using the first letter of each word from this phrase.

- Some of these passphrases might seem good and entirely unguessable, but it's easy to underestimate the capabilities of those invested in guessing passphrases.



- The words must actually be random. It might be tempting to make the passphrase make logical or grammatical sense, like “i enjoy blueberry pancakes” or “show me the money”, but this only makes your passphrase easier to guess. If your passphrase is likely to have ever been uttered in a movie, for instance, it’s probably on a list of cracked passwords somewhere, no matter how long it is.

- If you choose the words yourself, you are probably choosing from a shorter list of common words (~3,000) than exist in the entire English language (~170,000). This can make your passphrase easier to guess.

- Even if you attempt to be as random as possible in choosing the words in your own passphrase, you may unconsciously be stringing concepts together from pop culture or your personal life. The human brain just isn’t designed to be random!

- That’s why Diceware is such a powerful tool. It uses a real-life random number generator (dice) to implant a totally new, unique, password straight into your brain with zero paper trail.

- Our code generates an encrypted wordlist and uses the "Random" module which generates a 5-digit number as a simulation for rolling a die and then links it to a word in the word list. Code for the same is attached.

Dice Ware Passphrase Generator

```
n = int(input("\n Enter the number of words in the passphrase : "))
            dlist = list()
            directory = {}
            phrase = str()
            a = str()

            for x in range(n):

                for i in range(5):
                    a += str(random.randint(1, 6))  # randomly generate 5 digit
number.
                rdigit = a
                a = str()
                dlist.append(rdigit)
            print("\n")
            print(dlist)
```

Cryptography

- Encryption is the process of converting normal message (plaintext) into meaningless message (Ciphertext).
- Whereas Decryption is the process of converting meaningless message (Ciphertext) into its original form (Plaintext).
- cryptography is a package which provides cryptographic recipes and primitives to Python developers.


Encryption of passwords
from cryptography.fernet import Fernet
phrase = phrase.strip()
encPhrase = f.encrypt(phrase.encode())
decPhrase = f.decrypt(encPhrase).decode()

Program Output 1

```
Please Enter your Login Id { Name } : Aditya_Pai
Please Enter your Master Password : this is a sample passphrase

Hey there ! Please enter your choice to perform the desired
operation

1 Store an existing Password
2 Generate a password/passphrase
3 Retrieve a password
4 Calculate the strength of the password
5 Exit the program

Enter : 1

Enter the category against which the password will be stored :
facebook
Enter the password : roasted peanuts are unhealthy

Sit Back and relax, Your password is safe and secure in here ;)
```

Program Output 2

```
Enter : 2

 Type 1 for generating a password or Type 2 for generating a phrase
: 2

 Enter the number of words in the passphrase : 4


['21532', '65421', '36324', '44653']

Here is your Passphrase!

-->  crusher vaporizer lullaby portion  <--
```

System Architecture

1.Start

2. Ask the user for login id.

2.1 If login id in database, asks for password and checks if it is correct or not.
2.1.1 If password is incorrect ; then exits program ; else enters in the main program.

2.2 If login Id is not in data base asks for signing up
2.2.1 If user wants to sign up enters password in db and enters in the main program.
2.2.2 If user does not wish to sign up the program exits.

3.1 If user enters 1. Performs the function of storing password.

3.2 If user enters 2. Performs the function of generating passcode by asking 2 options : password or passphrase

3.2.1 If user enters 1. Performs the function of generating a password.

3.2.2 If user enters 2. Performs the function of generating a passphrase.

3.3 If user enters 3. Performs the function of retrieving a previously saved password.

3.4 If user enters 4. Performs the function of Determining the strength of an entered password.

3.5 exits the while loop.

4. End

Program Output 3

```
Enter : 3

Enter the category against which the password might have got stored
: facebook

User_ID    Category            Passcode            Timestamp
ADITYA_PAI FACEBOOK roasted peanuts are unhealthy 2022-07-01
```

Program Output 4

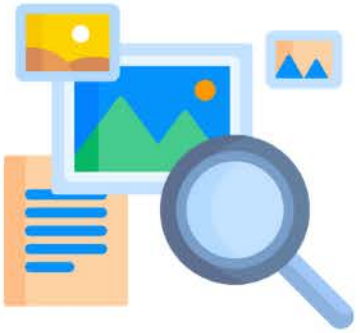
```
Enter : 4

Enter the password to check its strength ! : mustard exalted heave
shawl
Entropy :  126.91187238980947

Wow , This password is Extremely Strong !

Number of guesses :  800295545426922633557329006250866769 92

It will take a mammoth  25729 quadrillion years !!!!!
```

Program Output 5

```
Enter : 5

Process finished with exit code 0
```

References

- https://www.eff.org/deeplinks/2016/07/new-wordlists-random-passphrases
- https://diceware.dmuth.org/
- https://www.security.org/how-secure-is-my-password/
- https://redkestrel.co.uk/articles/random-password-strength/
- https://www.techtarget.com/whatis/definition/password-entropy#:~:text=Password%20entropy%20is%20a%20measurement,as%20well%20as%20password%20length
- https://www.geeksforgeeks.org/password-entropy-in-cryptography/
- https://www.omnicalculator.com/other/password-entropy
- https://gcn.com/cybersecurity/2020/09/a-computer-can-guess-more-than-1000000000000-passwords-per-second-still-think-yours-is-secure/315643/



Project Outcomes

In this mini project ,

- I used cryptography module , SqLite module , random module
- I built the logic for generating a random password and a passphrase.
- I understood the concept of encryption and used it to secure passwords in a database
- I learnt more about information theory and used it to calculate the entropy of the password.
- I then used this knowledge to embed it into a menu driven program and generated a desired output successfully :)

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Somaiya TRUST