PASSWORD ANALYZER

PRESENTED BY RIM FERCHICHI

IT 360

CONTENT

PROLEMATIC

Underestimating the power of hackers and the evolution of hacking methods.

Every once in a while when a company is hacked passwords are released and usually we use the same passwords for every login.

SOLUTION

Check if the password is commun

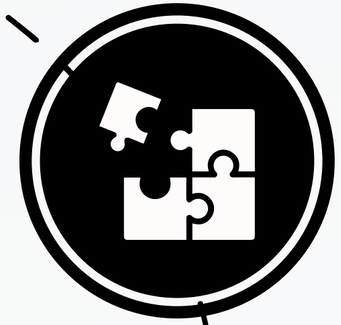
In your browser cache or in a worldwide commun list

Evaluate the strength

length / characters / numbers...

Suggest a password

according to words the user suggests



CODE : STRENGTH

Conditioning on different creteria : length / upper case / lower case / special / digits

```
uc=0
lc=0
p=0
d=0
t=0
for c in pwd :
    if c in string.ascii_uppercase :
        uc=uc+1
if uc>2 :
    uc=2

for c in pwd :
    if c in string.ascii_lowercase :
        lc=lc+1
if lc>2 :
    lc=2

for c in pwd :
    if c in string.punctuation :
        p=p+1
if p>2 :
    p=2

for c in pwd :
    if c in string.digits :
        d=d+1
if d>2 :
```

```
    d=2

if len(pwd)<4 :
    l=1
elif len(pwd)>=4 and len(pwd)<8 :
    l=4
elif len(pwd)>=8 and len(pwd)<12 :
    l=8
elif len(pwd)>=12 and len(pwd)<16 :
    l=10
else :
    l=12

t=l+uc+lc+p+d

if t < 5 :
    s="Your password is weak !"
    fill = '#bf211e'
elif t>=5 and t<10 :
    s="Your password is okay.."
    fill= '#ffa200'
elif t>=10 and t<15 :
    s="Your password is pretty good."
    fill='#003f88'
else :
```

CODE : SUGGESTION

Shuffling lists : list of words / upper case of the list / punctuation / digits.

```
def password_suggest():
    pwd_sg.config(state=NORMAL)
    word_list = str(entry_3.get()).split()
    word_list_upper = []
    for word in word_list :
        word_list_upper.append(word.upper())

    password_list = []
    punc = ["/","@",".","_","-","*","?","!"]
```

```
for i in range(6) :
    pwd=[]
    pwd.append(random.sample(word_list,1)[0])
    pwd.append(random.sample(punc,1)[0])
    pwd.append(random.sample(string.digits,1)[0])
    pwd.append(random.sample(word_list_upper,1)[0])
    pwd.append(random.sample(string.digits,1)[0])
    pwd.append(random.sample(string.digits,1)[0])
    pwd.append(random.sample(punc,1)[0])

    s=''.join([str(item) for item in pwd])
    password_list.append(s)

# canvas.itemconfig(tagOrId=pwd_sg, text="\n".join(map(str,password_list)))
# pwd_sg.delete(0, "end")
pwd_sg.insert("1.0", "\n".join(map(str,password_list)))
pwd_sg.config(state=DISABLED)
```

CODE : PASSWORD COLLECTION

Accessing local browser files to get the decrypted encryption key

key -----> encrypted (key) ----> encrypted (key)+DPAPI ---->encrypted (encrypted (key)+DPAPI)

```python
def get_encryption_key():
    local_state_path = os.path.join(os.environ["USERPROFILE"],
                                    "AppData", "Local", "Google", "Chrome",
                                    "User Data", "Local State")
    with open(local_state_path, "r", encoding="utf-8") as f:
        local_state = f.read()
        local_state = json.loads(local_state)
        # decode the encryption key from Base64
        key = base64.b64decode(local_state["os_crypt"]["encrypted_key"])
        # remove DPAPI str
        key = key[5:]
        # return decrypted key that was originally encrypted
        # using a session key derived from current user's logon credentials
        # doc: http://timgolden.me.uk/pywin32-docs/win32crypt.html
        return win32crypt.CryptUnprotectData(key, None, None, None, 0)[1]
```

CODE : PASSWORD COLLECTION

Decryption function.

CIPHER = F(**iv, key**)
Password = T(**CIPHER , encrypted(password)**)

```
def decrypt_password(password, key):
    try:
        # get the initialization vector
        iv = password[3:15]
        password = password[15:]
        # generate cipher
        cipher = AES.new(key, AES.MODE_GCM, iv)
        # decrypt password
        return cipher.decrypt(password)[:-16].decode()
    except:
        return str(win32crypt.CryptUnprotectData(password, None, None, None, 0)[1])
```

CODE : PASSWORD COLLECTION

Password collection and decryption function.

```python
def pass_collector(output_file):
    # get the AES key
    key = get_encryption_key()
    # local sqlite Chrome database path
    db_path = os.path.join(os.environ["USERPROFILE"], "AppData", "Local",
                           "Google", "Chrome", "User Data", "Profile 1", "Login Data")
    # copy the file to another location
    # as the database will be locked if chrome is currently running
    filename = "ChromeData.db"
    shutil.copyfile(db_path, filename)
    # connect to the database
    db = sqlite3.connect(filename)
    cursor = db.cursor()
    # `logins` table has the data we need
    cursor.execute("select username_value, password_value from logins order by date_created")

    # iterate over all rows
    with open(output_file, 'w') as f:
        for row in cursor.fetchall():
            username = row[0]
            password = decrypt_password(row[1], key)
            if username or password:
                f.write(f"{password}\n")
            else:
                continue

    cursor.close()
```

FUTURE POSSIBILITIES

- Migrate to Kali Linux
- Use predefined program Cupp to generate a list of the most likely passwords.
- Try to brute force it and show how much time it takes for a hacker to brute force your password.

THANK YOU FOR YOUR ATTENTION!