ЛАБОРАТОРНАЯ РАБОТА №5

ЗАЩИТА ДАННЫХ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

ВЫПОЛНИЛ студент группы ВПР31 Кононов Дмитрий

ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:

Разработать программу на языке высокого уровня типа, которая обеспечивает разграничение доступа при её запуске.

КОД ПРОГРАММЫ:

Python:

import math

from random import choice

import time

class File:

def \_\_init\_\_(self):

self.path = 'accounts.cvs'

def write(self, text):

with open(self.path, 'w') as f:

f.write(text)

def read(self):

with open(self.path) as f:

return f.read()

def append(self, login, password, perfect, deviation):

with open(self.path, 'a') as f:

f.write('\n{},{},{},{}'.format(login, password, perfect, deviation))

def get\_users(self):

with open(self.path) as f:

users = {}

for i in f:

user = i.split('\n')[0].split(',')

users[user[0]] = [user[1], float(user[2]), float(user[3])]

return users

class Passwords:

passwords = ['UPJirn', 'PR2d7p', 'f5cv0q', 'CsSP5r', 'CbuK7L', 'kklpeK', 'gdqlo2', 'FEPJhC',

'WzOwQk', '9ZenTW', 'Fjjl2s', 'I6aGfN', '8m5LYR', 'JTiSRr', 'wjmJhm', '37FTVt',

'6lj3TS', '5kLtEm', 'OoESOo', 'TjhAxl']

def \_\_write(self):

with open('passwords', 'w') as f:

string = ''

for i in self.passwords:

string += i + ','

f.write(string)

def \_\_read(self):

with open('passwords') as f:

self.passwords = []

for i in f.read().split(',')[:-1]:

self.passwords.append(i)

def get\_password(self):

return choice(self.passwords)

class User:

def \_\_init\_\_(self):

self.repeats = 4

self.eps = 2

def register(self):

login = input('Enter a login: ')

new\_password = Passwords().get\_password()

run = True

times = []

while run:

run = False

print('Enter password {} {} times:'.format(new\_password, self.repeats))

for i in range(self.repeats):

start = time.time()

password = input('Attempt {}, enter a password: '.format(i + 1))

times.append(time.time() - start)

if password != new\_password:

print('Uncorrected password, try again!')

run = True

times = []

break

File().append(login,

new\_password,

perfect := sum(times) / len(new\_password),

sum([abs(perfect - i) for i in times]) / len(new\_password))

print('{}! Registration has successful'.format(login))

def auth(self):

users = File().get\_users()

login = input('Enter a login: ')

if users.get(login):

start = time.time()

password = input('Enter a password: ')

wasted = time.time() - start

if not password == users.get(login)[0]:

print('Uncorrected password!')

return

else:

perfect = wasted \* 4 / len(password)

deviation = abs(wasted - users.get(login)[1]) \* 4 / len(password)

if not(abs(users.get(login)[1] - perfect) < self.eps and

abs(users.get(login)[2] - deviation) < self.eps):

print('Time mistake!')

else:

print('Successful')

else:

print('Uncorrected login!')

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

u = User()

while True:

a = input('1) Register new user.\n'

'2) Authorization user.\n'

'0) Exit.\n'

'Enter a number: ')

if a == '1':

u.register()

elif a == '2':

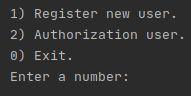
u.auth()

elif a == '0':

break

ПРИМЕР РАБОТЫ ПРОГРАММЫ

При запуске программы пользователю дается выбор: зарегистрироваться или войти под уже существующим аккаунтом.

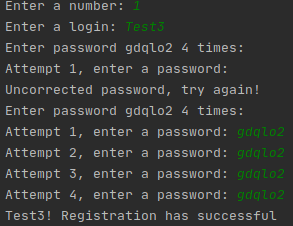


Выбираем пункт регистрация и проходим алгоритм регистрации.

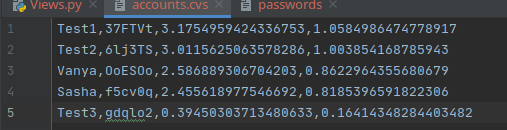
Пароль выбирается из заранее сгенерированного набора ключей.



Так выглядит регистрация нового пользователя.



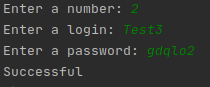
Регистрация прошла успешно, смотрим добавился ли пользователь в список зарегистрировавшихся.



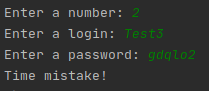
Пользователь добавился. Его логин, пароль, идеальное значение и отклонение.

Теперь пробуем войти.

При успехе будет следующее:



При сомнениях по времени со стороны программы произойдет следующее:



При неправильном пароле.

