МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра ЭВМ

Отчёт

Лабораторная работа № 1 по дисциплине

«Защита информации»

Выполнил студент группы ИВТб-4301\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Птахова А. М./

Проверил доцента кафедры ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Караваева О. В./

Киров 2024

1. Задание

Реализовать защиту от копирования

2. Ход работы

Вместе с основной программой поставляется программа активатор. Активация пользователя по средствам использования активатора заключается в записи имени и пароля пользователя в базу данных. Имя пользователя является уникальным идентификатором, поэтому перед записью данных в БД, происходит проверка наличия такого имени в базе. Кроме того, во время прохождения активации происходит привязка программы к компьютеру. Для выполнения такой привязки используется информация о компьютере (ОС, информация о процессоре), которая шифруется при помощи библиотеки cryptography с использованием хэш функции и в зашифрованном виде сохранятся в файл. Также создается и сохраняется сам ключ хэш функции в отдельном файле. Оба этих файла сохраняются в сжатый архив с паролем, который генерируется случайным образом. Далее этот пароль от архива также записывается в БД.

При выполнении основной программы первоначально происходит аутентификация пользователя при помощи ввода логина и пароля и проверкой введенных значений с БД. Следующий этап запуска программы – проверка данных компьютера, на которых запускается программа с теми, которые записаны в файле. Для этого происходит получение пароля от архива из БД, далее происходит извлечение файлов, получение информации из них. После получения информации происходит дешифрация информации, содержащий характеристики компьютера. После происходит сравнение текущей информации компьютера с информацией, записанной в файле. И на основании этого сравнения принимается дальнейшее действие – продолжение работы программы.

Для дополнительной защиты было реализовано автоматическое изменение пароля от архива через каждые 30 секунд, а также при запросе пользователя при выполнении основной программой.

Листинг кода приведен в приложении А. Экранные формы представлены в приложении Б.

3. Вывод

В ходе выполнения лабораторной работы были изучены методы защиты информации от копирования. Был изучен способ получения и использования информации об аппаратуре для реализации привязки программы к конкретному устройству. Был реализован способ защиты программы от копирования с использованием цифровой подписи и сервера активации

Приложение А

Листинг кода

Db.py

import postgresql

from postgresql import exceptions

class database:

\_connection = None

\_db\_url = "pq://postgres:74771@localhost:5432/lab\_securityInfo"

def openConnection(self):

try:

self.\_connection = postgresql.open(self.\_db\_url)

print("Подключено")

except postgresql.exceptions.AuthenticationMethodError: # неподдерживаемый тип авторизации

print("Ошибка! Неподдерживаемый тип авторизации.")

except postgresql.exceptions.ClientCannotConnectError: # клиенту не удалось установить соединение с сервером

print("Ошибка! Клиенту не удалось установить соединение с сервером.")

except postgresql.exceptions.ConnectTimeoutError: # клиенту не удалось установить соединение в заданное время

print("Ошибка! Клиенту не удалось установить соединение в заданное время.")

except postgresql.exceptions.DriverError: # Ошибка в реализации драйвера

print("Ошибка в реализации драйвера")

def getGroup(self):

try:

return self.\_connection.query("SELECT \* FROM public.student")

except postgresql.exceptions.ConnectionDoesNotExistError:

print("Error")

except:

print("Error1")

def chooseId(self, number):

try:

sql = "SELECT \* FROM public.users WHERE id ="+str(number)

ps = self.\_connection.query(sql)

return ps

except:

print("Error!")

def updateInfo(self, id\_person, password\_folder):

try:

sql = "update users set password\_folder="+str(password\_folder)+" where id="+str(id\_person)

self.\_connection.query(sql)

except:

print("Error!")

def addPerson(self, id\_person, password, password\_folder):

try:

val\_str = "("+str(id\_person) + ",'"+password+"', "+str(password\_folder)+")"

print(val\_str)

self.\_connection.query("INSERT INTO users(id, password, password\_folder) VALUES"+val\_str)

except:

print("Error")

def delPerson(self, id\_person):

try:

self.\_connection.query("DELETE FROM student where id="+str(id\_person))

except:

print("Error")

def closeConnection(self):

self.\_connection.close()

Main.py

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from db import database

from tkinter.messagebox import showwarning

import random

import platform

from cryptography.fernet import Fernet

import subprocess

import pyminizip

import os

def generateRandomNumber():

return random.randint(256, 541628)

def writeInfoToFile(filename, text, mode):

file = open(filename, mode)

file.write(text)

file.close()

def getSysInfo():

mach = platform.machine()

vers = platform.version()

proc = platform.processor()

text = mach + vers + proc

text\_decod = text.encode('utf-8')

writeInfoToFile('test2.txt', text\_decod, 'wb')

def generateKey():

key = Fernet.generate\_key()

f = Fernet(key)

writeInfoToFile("test.txt",key, 'wb')

def decryptFile():

with open ('test.txt', 'rb') as keyfile:

key = keyfile.read()

with open('test2.txt', 'rb') as file:

f = Fernet(key)

text = file.read()

dec\_text = f.decrypt(text)

print(dec\_text.decode())

def encryptFile():

with open('test.txt', 'rb') as filekey:

key = filekey.read()

with open("test2.txt", "rb") as file:

f = Fernet(key)

text = file.read()

enc\_text = f.encrypt(text)

writeInfoToFile("test2.txt",enc\_text, 'wb')

def createZipFile(gener\_password):

pyminizip.compress\_multiple([u'test.txt', 'test2.txt'], [u'', u''], "file.zip", str(gener\_password), 4)

def changePassword():

createZipFile()

def clearDir():

os.remove('test.txt')

os.remove('test2.txt')

class App(tk.Tk):

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

super().\_\_init\_\_()

opts={'ipadx':4, 'ipady':4, 'sticky':'nswe'}

label\_id = tk.Label(self, text= "Name:")

label\_id.grid(row=0, column = 1, \*\*opts)

edit\_id = ttk.Entry()

edit\_id.grid(row=0, column=2, columnspan=1, \*\*opts)

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=1, column=0, columnspan=4, \*\*opts)

label\_passw = tk.Label(self, text="Password")

label\_passw.grid(row=2, column =1, \*\*opts)

edit\_passw = tk.Entry()

edit\_passw.grid(row=2, column =2, columnspan=1, \*\*opts)

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=3, column=0, columnspan=4, \*\*opts)

display\_text = tk.StringVar()

label\_info = tk.Label(self, textvariable=display\_text)

label\_info.grid(row=6, rowspan=2, column=0, columnspan=4, \*\*opts)

def add\_str():

display\_text.set("")

number\_id = edit\_id.get()

password = edit\_passw.get()

password\_folder = generateRandomNumber()

dtbs = database()

dtbs.openConnection()

s = dtbs.chooseId(number\_id)

print(len(s))

if (s):

print("found")

display\_text.set("This name is already exist. Change name")

else:

s = dtbs.addPerson(number\_id, password, password\_folder)

display\_text.set("Registration is gone succesfully")

generateKey()

getSysInfo()

encryptFile()

decryptFile()

createZipFile(password\_folder)

clearDir()

dtbs.closeConnection()

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=5, column=0, columnspan=4, \*\*opts)

btn\_add=ttk.Button(text="Registration", command = add\_str)

btn\_add.grid(row=4, column=1,columnspan =2,\*\*opts)

if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.title('Регистрация')

app.mainloop()

Prog.py

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from db import database

from tkinter.messagebox import showwarning

import random

import platform

from cryptography.fernet import Fernet

import subprocess

import pyminizip

import os

from main import clearDir, generateRandomNumber, createZipFile

class App(tk.Tk):

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

super().\_\_init\_\_()

opts={'ipadx':4, 'ipady':4, 'sticky':'nswe'}

label\_id = tk.Label(self, text= "Name:")

label\_id.grid(row=0, column = 1, \*\*opts)

edit\_id = ttk.Entry()

edit\_id.grid(row=0, column=2, columnspan=1, \*\*opts)

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=1, column=0, columnspan=4, \*\*opts)

label\_passw = tk.Label(self, text="Password")

label\_passw.grid(row=2, column =1, \*\*opts)

edit\_passw = tk.Entry()

edit\_passw.grid(row=2, column =2, columnspan=1, \*\*opts)

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=3, column=0, columnspan=4, \*\*opts)

display\_text = tk.StringVar()

label\_info = tk.Label(self, textvariable=display\_text)

label\_info.grid(row=6, rowspan=2, column=0, columnspan=4, \*\*opts)

def getSysInfo():

mach = platform.machine()

vers = platform.version()

proc = platform.processor()

text = mach + vers + proc #type = str

checkActivation(text.encode('utf-8'))

def openFile(filename):

try:

file = open(filename, 'rb')

except FileNotFoundError:

display\_text.set('oops, filenotFound')

return ""

else:

return file.read()

def checkActivation(text\_):

if (openFile('test.txt') != "" and openFile('test2.txt') != ""):

key = openFile('test.txt')

f = Fernet(key)

text = openFile('test2.txt')

dec\_text = f.decrypt(text)

convert\_text = dec\_text

print(convert\_text)

print(text\_)

if (convert\_text == text\_):

display\_text.set('success')

clearDir()

else:

display\_text.set('oops, sometimes going wrong')

def changePassword(str\_pass\_folder):

pyminizip.uncompress("file.zip", str\_pass\_folder, "", 0)

password\_folder = generateRandomNumber()

createZipFile(password\_folder)

return password\_folder

def add\_str():

display\_text.set("")

number\_id = edit\_id.get()

password = edit\_passw.get()

dtbs = database()

dtbs.openConnection()

s = dtbs.chooseId(number\_id)

if (s):

for data in s:

if (data[1]==password):

pass\_folder = str(data[2])

change\_passw = changePassword(pass\_folder)

dtbs.updateInfo(number\_id, change\_passw)

display\_text.set("It's okay")

else:

display\_text.set("Name or password is wrong")

else:

display\_text.set("You are haven't registration")

dtbs.closeConnection()

dtbs = database()

dtbs.openConnection()

s = dtbs.chooseId(number\_id)

if (s):

for data in s:

if (data[1]==password):

str\_pass\_folder = str(data[2])

try:

pyminizip.uncompress("file.zip", str\_pass\_folder, "", 0)

getSysInfo()

except:

display\_text.set("Password was changed")

else:

display\_text.set("Name or password is wrong")

else:

display\_text.set("You are haven't registration")

dtbs.closeConnection()

label\_otstup = tk.Label(self, text='')

label\_otstup.grid(row=5, column=0, columnspan=4, \*\*opts)

btn\_add=ttk.Button(text="Log out", command = add\_str)

btn\_add.grid(row=4, column=1,columns

pan =2,\*\*opts)

Приложение Б

Экранные формы

