Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский Университет ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Разработка систем аутентификации и криптографии»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 Разработка системы парольной аутентификации в клиент-сервером приложении**

**«Вариант 5 - Реализация аутентификации по паролю с подтверждением по email»**

**Выполнил:**

Магистрант гр. N42514c

В.В. Герасимов



**Проверил:**

И.Р. Фёдоров

Санкт-Петербург

2020 г.

**Содержание**

[1. Цель работы 3](#_Toc56447885)

[2. Описание выбранных средств реализации и обоснования выбора 3](#_Toc56447886)

[3. Описание алгоритма 4](#_Toc56447887)

[4. Исходный код и демонстрация работы программы 4](#_Toc56447888)

[4.1. Демонстрация работа программы 11](#_Toc56447889)

[5. Выводы 11](#_Toc56447890)

1. Цель работы

Лабораторная работа №2:

Требования к реализации программы:

* необходимо реализовать метод аутентификации в клиент-серверном приложении согласно варианту
* клиент должен представлять собой веб-страницу с формой авторизации пользователя
* сервер должен включать в себя две части:

1) таблица идентификаторов (данные о пользователях для аутентификации: логин/пароль/токен/и т. д. в зависимости от метода аутентификации);

2) процесс с реализованной логикой метода аутентификации.

1. Описание выбранных средств реализации и обоснования выбора

Для реализации алгоритма был выбран язык Python, так как к скорости выполнения программы не предъявлялось каких-либо особых требований.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Этот язык имеет довольно простой синтаксис, лёгок в изучении и имеет низкий порог вхождения. В то же время язык является очень мощным средством разработки. Такой выбор также обусловлен тем, что это достаточно распространённый язык в настоящее время.

Для реализации клиент-серверной архитектуры был выбран фреймворк Flask.

1. Описание алгоритма

В таблице идентификаторов должны хранятся: логин, email, пароль, хеш временного кода подтверждения (MD5). Таблица идентификаторов представляет собой таблицу в реляционной БД, данные передаются через SQL-запросы. При аутентификации на сервере сравниваются пароли и на email пользователя отправляется сгенерированный на сервере временный код подтверждения. На клиенте после отправки данных с паролем происходит редирект на форму для ввода временного кода подтверждения. После отправки кода на сервере сравниваются хеш пришедшего кода и хеш кода из БД (MD5). При совпадении хешей аутентификация считается успешной и происходит редирект на страницу-заглушку.

1. Исходный код и демонстрация работы программы

[Ссылка на исходный код](https://github.com/gevol69/dev_auth_crypt_system/tree/main/dev_auth_crypt_system_lab_2)

Файл – **main.py**:

**from** flask **import** Flask, render\_template, request, redirect, url\_for, session

**import** models **as** dbHandler

**from** random **import** randint

**import** hashlib

**from** flask\_mail **import** Mail, Message

**from** decorators **import** async\_func

app = Flask(\_\_name\_\_)

app.secret\_key = '73870e7f-634d-433b-946a-8d20132bafac'

app.config['MAIL\_SERVER'] = 'smtp.googlemail.com'

app.config['MAIL\_PORT'] = 465

app.config['MAIL\_USE\_TLS'] = False

app.config['MAIL\_USE\_SSL'] = True

app.config['MAIL\_USERNAME'] = 'authsystemtest@gmail.com' # введите свой адрес электронной почты здесь

app.config['MAIL\_DEFAULT\_SENDER'] = 'authsystemtest@gmail.com' # и здесь

app.config['MAIL\_PASSWORD'] = '123QWEasd!' # введите пароль

mail = Mail(app)

@app.route('/temp\_code', methods=['POST', 'GET'])

**def** check\_code():

**if** session.get('data')['login'] == 'Process':

username = session.get('data')['username']

temp\_code\_bd = dbHandler.get\_temp\_code(username)[0][0]

**if** request.method=='POST':

temp\_code\_from\_user = request.form['temp\_code']

temp\_code\_from\_user\_md5 = str(hashlib.md5(str(temp\_code\_from\_user).encode()).hexdigest())

**if** temp\_code\_bd == temp\_code\_from\_user\_md5:

session['data'] = dict(username=username, login='True')

**return** redirect(url\_for('success'))

**else**:

**return** render\_template('temp\_code.html', tempcodefalse=True)

**else**:

**return** render\_template('temp\_code.html')

**else**:

**return** redirect(url\_for('login'))

@app.route('/success')

**def** success():

**if** session.get('data')['login'] == 'True':

username = session['data']['username']

**return** render\_template('success.html', name=username)

**else**:

**return** redirect(url\_for('login'))

@async\_func

**def** send\_async\_email(msg):

**with** app.app\_context():

mail.send(msg)

**def** generate\_and\_send\_temp\_code(username, email):

temp\_code = randint(10000, 99999)

hash\_temp\_code = str(hashlib.md5(str(temp\_code).encode()).hexdigest())

#send code email

msg = Message("Код подтверждения", recipients=[email])

msg.body = "�'аш код подтверждения - {}".format(temp\_code)

send\_async\_email(msg)

**return** hash\_temp\_code

@app.route('/', methods=['POST', 'GET'])

**def** login():

session['data'] = dict(login='False')

**if** request.method=='POST':

username = request.form['username']

password = request.form['password']

user = dbHandler.retrieveUsers(username)

**if** user:

**if** password == user[0][1]:

email = user[0][2]

md5\_hash\_temp\_code = generate\_and\_send\_temp\_code(username, email)

dbHandler.insert\_temp\_code(username, md5\_hash\_temp\_code)

session['data'] = dict(username=username, login='Process')

**return** redirect(url\_for('check\_code'))

**else**:

**return** render\_template('index.html', Passwordfalse=True)

**else**:

**return** render\_template('index.html', loggedfalse=True)

**else**:

**return** render\_template('index.html')

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

app.run(debug=True)

Файл – **decorators.py**:

|  |  |
| --- | --- |
|  | **from** threading **import** Thread  **def** async\_func(f):  **def** wrapper(\*args, \*\*kwargs):  thr = Thread(target = f, args = args, kwargs = kwargs)  thr.start()  **return** wrapper |

Файл – **models.py**:

|  |  |
| --- | --- |
|  | **import** sqlite3 **as** sql  **def** retrieveUsers(username):  con = sql.connect("database.db")  cur = con.cursor()  cur.execute("SELECT username, password, email FROM users WHERE username = '{}'".format(username))  users = cur.fetchall()  con.close()  **return** users  **def** insert\_temp\_code(username, hash\_temp\_code):  con = sql.connect("database.db")  cur = con.cursor()  sql\_query = "UPDATE users SET md5\_temp\_code = '{}' WHERE username = '{}'".format(hash\_temp\_code, username)  cur.execute(sql\_query)  con.commit()  con.close()  **def** get\_temp\_code(username):  con = sql.connect("database.db")  cur = con.cursor()  cur.execute("SELECT md5\_temp\_code FROM users WHERE username = '{}'".format(username))  md5\_temp\_code = cur.fetchall()  con.close()  **return** md5\_temp\_code |

Файл – **success.html**:

<html lang="en">

<head>

<link href="static/main.css" rel="stylesheet" type="text/css" media="all">

</head>

<body>

<h1 style="text-align:center;">Аутентификация прошла успешно!</h1>

<h1 style="text-align:center;">Hello, {{ name }}!</h1>

</body>

</html>

Файл – **index.html**:

<html lang="en">

<head>

<link href="static/main.css" rel="stylesheet" type="text/css" media="all">

</head>

<body>

<h1 style="text-align:center;">Парольная аутентификация</h1>

<div class="box">

<h1>�'ход</h1>

<form action="/" method="post">

<input type="text" name="username" id="username" placeholder="�мя пользователя" required/>

<br>

<input type="password" name="password" id="password" placeholder="Пароль" required/>

<br>

<input type="submit" class="button" value="�'ойти"/>

</form>

</div>

{%if loggedfalse%}

<div class="Login False" style="margin-left:20%">

<h2>Такого пользователя не существует!</h2>

</div>

{%endif%}

{%if Passwordfalse%}

<div class="Password False" style="margin-left:20%">

<h2>�'ы ввели неправильный пароль!</h2>

</div>

{%endif%}

</body>

</html>

Файл – **temp\_code.html**:

<html lang="en">

<head>

<link href="static/main.css" rel="stylesheet" type="text/css" media="all">

</head>

<body>

<h1 style="text-align:center;">�'ременный код отправлен вам на email</h1>

<div class="box">

<h1>�'ведите код</h1>

<form action="/temp\_code" method="post">

<input type="text" name="temp\_code" id="temp\_code" placeholder="�'ведите временный код" required/>

<br>

<input type="submit" class="button" value="Отправить"/>

</form>

</div>

{%if tempcodefalse%}

<div class="Temp Code False" style="margin-left:20%">

<h2>�'ы ввели неправильный код подтверждения, повторите попытку!</h2>

</div>

{%endif%}

</body>

</html>

Файл – **main.css**:

body,

input,

textarea,

select {

font-family: 'Open Sans', sans-serif;

font-size: 14px;

color: #4c4c4c;

}

h1 {

font-size: 32px;

font-weight: 300;

color: #4c4c4c;

text-align: center;

padding-top: 10px;

margin-top: 20px;

margin-bottom: 10px;

}

.box h2{

text-align: center;

margin-top: 30px;

}

.box {

margin: 20px auto;

width: 310px;

height: 260px;

-webkit-border-radius: 8px/7px;

-moz-border-radius: 8px/7px;

border-radius: 8px/7px;

background-color: white;

-webkit-box-shadow: 1px 2px 5px grey;

-moz-box-shadow: 1px 2px 5px grey;

box-shadow: 1px 2px 5px grey;

border: solid 1px grey;

}

.footer h4{

text-align: right;

margin-right: 30%;

margin-top: 30px;

}

input{

margin-top: 15px;

margin-left: 25%;

margin-right:25%;

}

* 1. Демонстрация работа программы

Работа программы продемонстрирована в [этой](http://ipic.su/img/img7/fs/bandicam2020-11-1619-20-45-565.1605543905.gif) демке.

1. Выводы

В ходе лабораторной работы разработана система парольной аутентификации с подтверждением по email в клиент-сервером приложении.