哈爾濱工業大學

网络安全实验报告

题　　目　认证实验

专　　业　信息安全

学　　号　2021111000

学　　生　李卓凌

指 导 教 师　王彦

一、实验目的

实现基于口令的客户端与服务端的认证过程 。

二、实验内容

1. 服务端建立服务（终端、网页等），对客户端发送的请求进行响应。

2. 客户端对服务端发送请求，对服务端响应的数据进行处理。

3. 完成注册、登录、修改密码的功能；

注册：客户端输入用户名、口令，使用散列函数计算用户名与口令的散列值 1，将用户名、散列值 1 明文发送到服务端。服务端存储至数据库；

登录：客户端输入用户名，口令，随机产生认证码，使用散列函数计算用户名与口令的散列值 1，使用散列值 1 与认证码计算散列值 2，将用户名，散列值 2，认证码明文传送到服务器端。接收服务端响应的信息，如果登录成功，解密出认证码存储至本地文件。若登录失败，打印出错误信息；服务端利用数据库存储的散列值 1 与请求数据里的认证码计算散列值 2’，与客户端发送的请求数据散列值 2 做对比，相等则登录成功，服务端将认证码用散列值 1 加密，发送给客户端。若登录失败，返回错误信息；

修改密码：客户端输入用户名、旧口令、新口令，使用散列函数计算用户名与旧口令的散列值 1、用户名于新口令的散列值 1’，将用户名与两个散列值发送给服务端。服务端接收数据，将旧散列值 1 与数据库存储的散列值 1 做比对，若相等，
将新散列值 1 覆盖存储至响应用户名处；若对比失败，返回错误信息。

三、实验过程

实验环境：

Windows10； Py Charm

1. 客户端

哈希函数：

```
2个用法
def hash_sha256(data):
    return hashlib.sha256(data.encode()).hexdigest()
```

认证函数：

```
1个用法
def authenticate(username, password):
    nonce = str(random.randint( a: 100000,  b: 999999))
    print(nonce)
    hash1 = hash_sha256(username + password)
    hash2 = hash_sha256(hash1 + nonce)

    data = {
        'username': username,
        'hash2': hash2,
        'nonce': nonce
    }

    response = requests.post( url: 'http://127.0.0.1:5000/authenticate', json=data)
    if response.status_code == 200:
        encrypted_nonce = response.json().get('encrypted_nonce')
        # 解密 encrypted_nonce
        decrypted_nonce = decrypt_nonce(encrypted_nonce, hash1)
        # 将 decrypted_nonce 写入文件
        with open('auth_code.txt', 'w') as file:
            file.write(decrypted_nonce)
        print("Authentication successful, auth code written to file.")
    else:
        print("Authentication failed.")
```

修改密码：

```
def change_password(username, old_password, new_password):
    data = {
        'username': username,
        'old_password': old_password,
        'new_password': new_password
    }

    response = requests.post( url: 'http://127.0.0.1:5000/change_password', json=data)
    if response.status_code == 200:
        print("Password changed successfully.")
    else:
        print("Password change failed:", response.json().get('error'))

# 示例
username = input("Enter username: ")
password = input("Enter password: ")

authenticate(username, password)

change_choice = input("Do you want to change your password? (yes/no): ")
if change_choice.lower() == 'yes':
    old_password = password
    new_password = input("Enter new password: ")
    change_password(username, old_password, new_password)
```

2. 服务端

使用flask 框架快速创建网页服务，使用MySQL 数据库存储数据：本地创建数据库authenlab。

```python
# 数据库配置
db_config = {
    'user': 'root',
    'password': 'root',
    'host': 'localhost',
    'database': 'auth_db'
}


2 个用法
def get_db_connection():
    return mysql.connector.connect(**db_config)
```

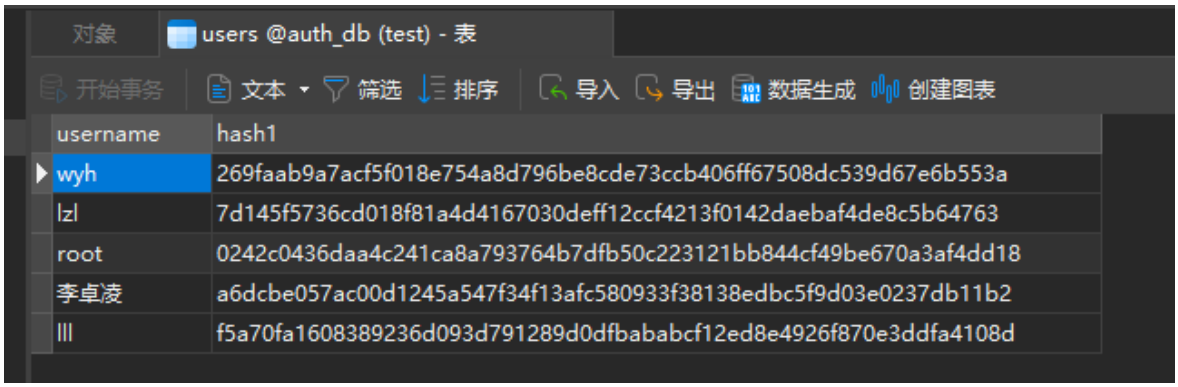
接受前端的请求：

```python
@app.route( rule: '/authenticate', methods=['POST'])
def authenticate():
    data = request.json
    username = data['username']
    hash2 = data['hash2']
    nonce = data['nonce']

    conn = get_db_connection()
    cursor = conn.cursor()

    cursor.execute( operation: "SELECT hash1 FROM users WHERE username = %s", params: (username,))
    row = cursor.fetchone()
    cursor.close()
    conn.close()

    if row:
        hash1 = row[0]
        hash2_prime = hashlib.sha256((hash1 + nonce).encode()).hexdigest()

        if hash2 == hash2_prime:
            # 加密 nonce
            key = generate_key_from_hash1(hash1)
            fernet = Fernet(key)
            encrypted_nonce = fernet.encrypt(nonce.encode()).decode()
            return jsonify({'encrypted_nonce': encrypted_nonce})

    return jsonify({'error': 'Authentication failed'}), 401
```

```python
@app.route( rule: '/change_password', methods=['POST'])
def change_password():
    data = request.json
    username = data['username']
    old_password = data['old_password']
    new_password = data['new_password']

    old_hash1 = hashlib.sha256((username + old_password).encode()).hexdigest()
    new_hash1 = hashlib.sha256((username + new_password).encode()).hexdigest()

    conn = get_db_connection()
    cursor = conn.cursor()

    cursor.execute( operation: "SELECT hash1 FROM users WHERE username = %s", params: (username,))
    row = cursor.fetchone()

    if row and row[0] == old_hash1:
        cursor.execute( operation: "UPDATE users SET hash1 = %s WHERE username = %s", params: (new_hash1, username))
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({'status': 'Password changed successfully'})

    cursor.close()
    conn.close()
    return jsonify({'error': 'Old password is incorrect'}), 401
```

四、实验结果

首先查看数据库内容：已经存在先前注册好的用户。



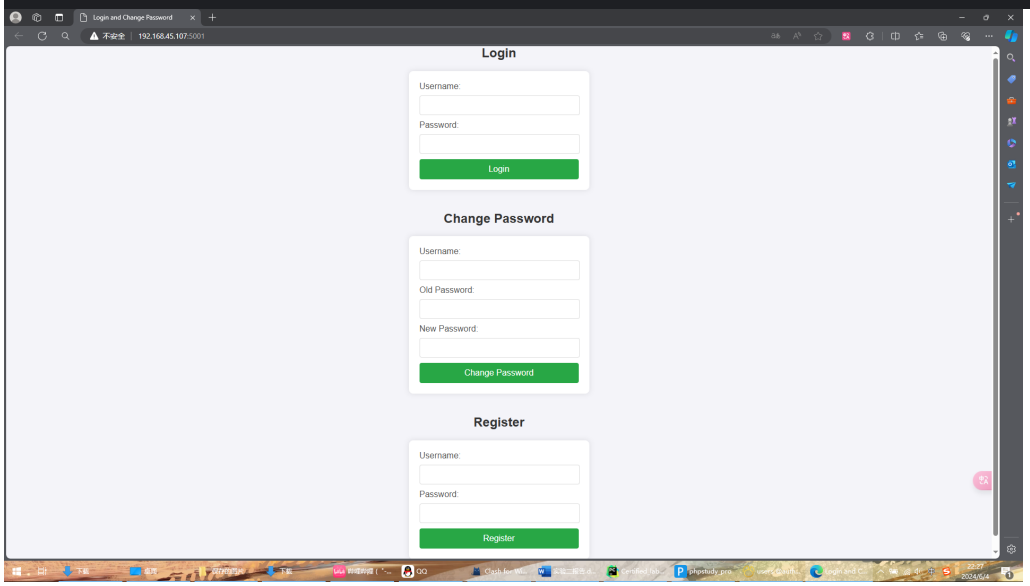
username	hash1
wyh	269faab9a7acf5f018e754a8d796be8cde73ccb406ff67508dc539d67e6b553a
lzl	7d145f5736cd018f81a4d4167030deff12ccf4213f0142daebaf4de8c5b64763
root	0242c0436daa4c241ca8a793764b7dfb50c223121bb844cf49be670a3af4dd18
李卓凌	a6dcbe057ac00d1245a547f34f13afc580933f38138edbc5f9d03e0237db11b2
lll	f5a70fa1608389236d093d791289d0dfbababcf12ed8e4926f870e3ddfa4108d

分别启动服务端：

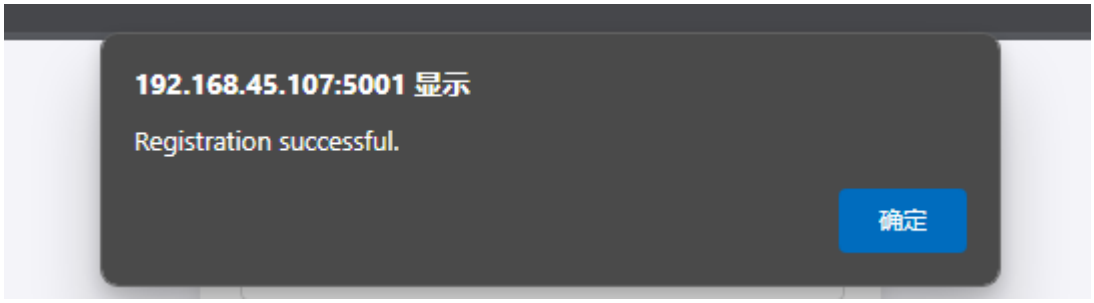
```
C:\Users\lzl\.conda\envs\py_demo\python.exe C:\Users\lzl\PycharmProjects\Certified_lab\server.py
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.45.107:5000
Press CTRL+C to quit
```

启动客户端：

```
C:\Users\lzl\.conda\envs\py_demo\python.exe C:\Users\lzl\PycharmProjects\Certified_lab\Test\web_client.py
 * Serving Flask app 'web_client'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.45.107:5001
Press CTRL+C to quit
```

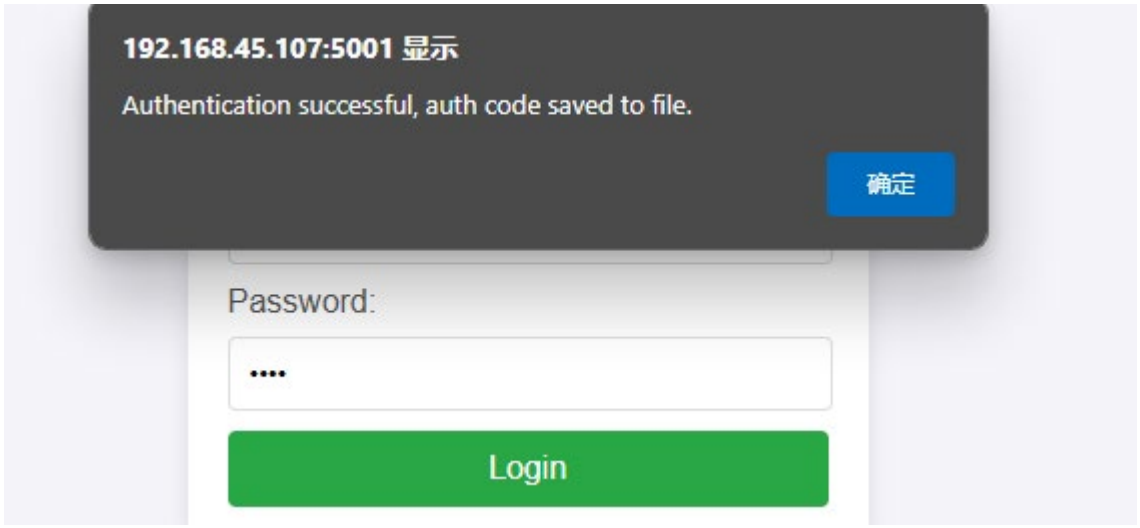


首先注册一个新用户：

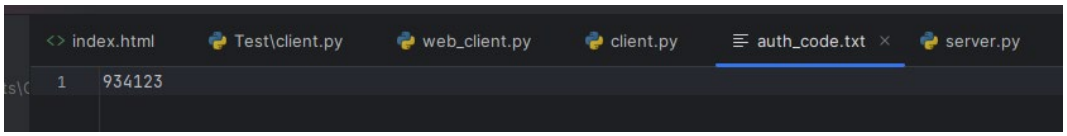


查看数据库，此时增加了刚刚注册的用户：

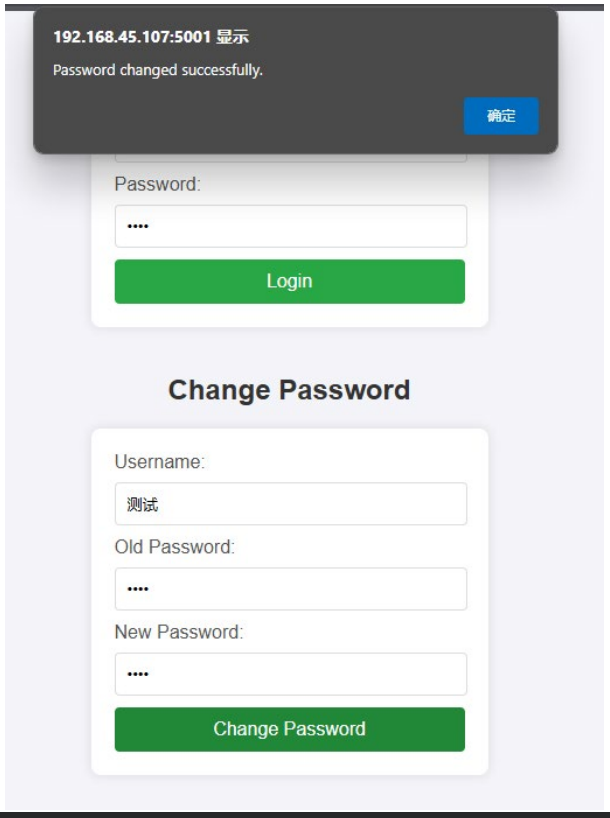
用户登录：



查看保存的code：



修改密码：



测试	cc7ff3bb51e84163980b69e93150f77e8510b49f9cc663810b1bae281848d16e

五、心得体会

经过此次实验，对基于口令的认证过程有了更多的认识。对随机函数、散列函数、加解密函数的使用更熟练。

此次实验多个地方由ChatGPT提供意见，给予感谢