## CS763 Lab 3 Password-based Authentication

Donghang He

### Compile and run the program

Because I didn’t use any database, so it is very easy to compile and run the program using IntelliJ IDEA.

文本

描述已自动生成

### Structure

图形用户界面, 文本, 应用程序

描述已自动生成

The structure of the entire project is shown in the figure above. The PasswordApplication class is the main class used to execute the entire project, and the database.txt file is used to store user information. The classes in the Basic package are all basic functions, including registration, login, password storage, and logout. The classes in the Operation package are operation functions, including checking registration input information, password encryption, file reading, and login information matching.

### Crypto API

1. PBKDF2

PBKDF2 (Password-Based Key Derivation Function) is a function used to derive keys and is often used to generate encrypted passwords. Its basic principle is to pass a pseudo-random function, take the plaintext and a salt value as input parameters, and then repeat the operation, and finally generate the key. If the number of repetitions is large enough, the cost of cracking will become very high. The addition of salt will also increase the difficulty of the attack.

1. PBKDF2 Function definition

DK = PBKDF2(PRF, Password, Salt, c, dkLen)

**PRF** is a pseudo-random function, in this project, SecureRandom is mainly used to generate salt

**Password** is the original password used to generate the key

**Salt** is a salt value for encryption

**c** is the number of repeated calculations, 1000 iterations used in this project

**dkLen** is the length of the expected key, use a length of 64\*8 in this project

**DK** is the last key generated

1. PBKDF2 Algorithm flow

The value of DK is formed by splicing more than one block. The number of blocks is the value of dkLen/hLen (length of salt). That is to say, if the PRF output result is shorter than the expected key length, then multiple results must be spliced to meet the key length:

DK = T1 || T2 || ... || Tdklen/hlen

And each block is obtained by function F:

Ti = F(Password, Salt, c, i)

In function F, PRF will perform c operations, and then XOR the result to get the final value

F(Password, Salt, c, i) = U1 ^ U2 ^ ... ^ Uc

For the first time, PRF will use Password as the key, and Salt concatenated with a 32-bit integer encoded in big-endian order as the salt value for operation.

U1 = PRF(Password, Salt || INT\_32\_BE(i))

The subsequent c-1 time will use the result obtained last time as the salt value.

### Source code

1. Sign up

After the system is running, when the user enters 1, it will switch to the registration function. Under this function, the user will be required to enter the username and password, and then enter the confirm password again.

1. Username password check style.

When the user enters the username, the system will automatically check whether it is qualified, and if it is not qualified, it will skip and let the user re-enter the username. The password will first check whether the two entries are the same, and then check whether it complies with the rules.

The inspection process of the two is basically the same, and both use the method under character to detect .

1. Password encryption

After the username and password are in the correct format, the password will be encrypted. The password encryption uses the PBKDF2 algorithm. First, you need to generate 16 as the salt value. Then perform 1000 iterations of encryption, and the result will be converted to hexadecimal. The salt value will also be converted to hexadecimal.

1. Information store in file

After getting the salt value and encrypted password, a new file will be created, and the information will be stored in the file.

1. Login

The user can enter the login function by entering 2 in the menu interface. The system will ask the user to enter a username and password.

1. Get password from file

The system reads the storage file and obtains the password and salt value according to the username.

1. Password match

Encrypt the password newly entered by the user and the salt value obtained from the storage, and compare the result obtained with the original encrypted password. Same as login success.

1. Log out

After logging in successfully, you can log out by typing exit.

图示

描述已自动生成

### Execution result

1. Menu

文本

描述已自动生成

1. Sign up

文本

描述已自动生成

1. Log in

文本

描述已自动生成

1. Log out

图形用户界面, 文本, 应用程序

描述已自动生成

### Summary

The main purpose of this project is to understand password encryption and the password matching mechanism when logging in. In this project, I learned a lot, including different password encryption mechanisms, password matching and so on. Although only one type of encrypted content is used, I still have a brief understanding of each.

In this project, the database is not used, because the main content of the project is password encryption. Therefore, a corresponding simplification was carried out.

There are still many areas to be improved in this project. For example, only single user registration and login are currently supported. You can use hashmap to store the user information in the file when it is read to support multiple users. Of course, using a database would be a better choice.

At the same time, there is no way to hide the password when executing a Java program in the IDE. Although the password can be hidden by entering it from the command line, it is not implemented because the project package needs to be compiled one by one. It can be modified to log in using a webpage, and it can be realized by changing to a web app. The entire project document is clearly described and the difficulty is moderate.