

به نام خدا



دانشگاه اصفهان
University of Isfahan

پروژه پایانترم درس مبانی امنیت سایبری تاریخ: 1402/11/8

توسط: ابراهیم کیانی فلاورجانی

شماره دانشجویی: 993623036

شرح پروژه:

پروژه شامل سه فایل است. یک فایل شامل یک دیتابیس خالی، و دو فایل دیگر به زبان پایتون است.

یک فایل پایتون به عنوان سرور و فایل دیگر به عنوان بخش کلاینت در نظر گرفته شده اند. هر دو فایل از برنامه نویسی سوکت پایتون بهره مندی شده اند. و زمانی که این فایل ها اجرا شوند، آنها از طریق آپی و پورت مشخص شده در هر فایل به یکدیگر متصل شده اند. فایل سرور از دیتابیس مشخص شده برای ذخیره نام کاربری و رمز کاربر استفاده میکند.

من در هر دو فایل رمز کاربر را عدد 1234 به صورت قرار دادی گذاشته ام که مثلاً قبلاً توسط کاربر و سرور در یک کانال امن رد و بدل شده است.

برای اجرای پروژه ابتدا فایل سرور در یک ترمینال باید اجرا شود به صورت زیر:

```
C:\Windows\system32\cmd.exe - python server.py
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ebrahim Kiani>cd OneDrive\Desktop\security_project

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>python server.py
Server listening on 127.0.1.1:12345
```

اکنون فایل کلاینت را در ترمینال دیگر باز میکنیم:

```
C:\Windows\system32\cmd.exe - python client.py

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>python client.py
please enter your username:
```

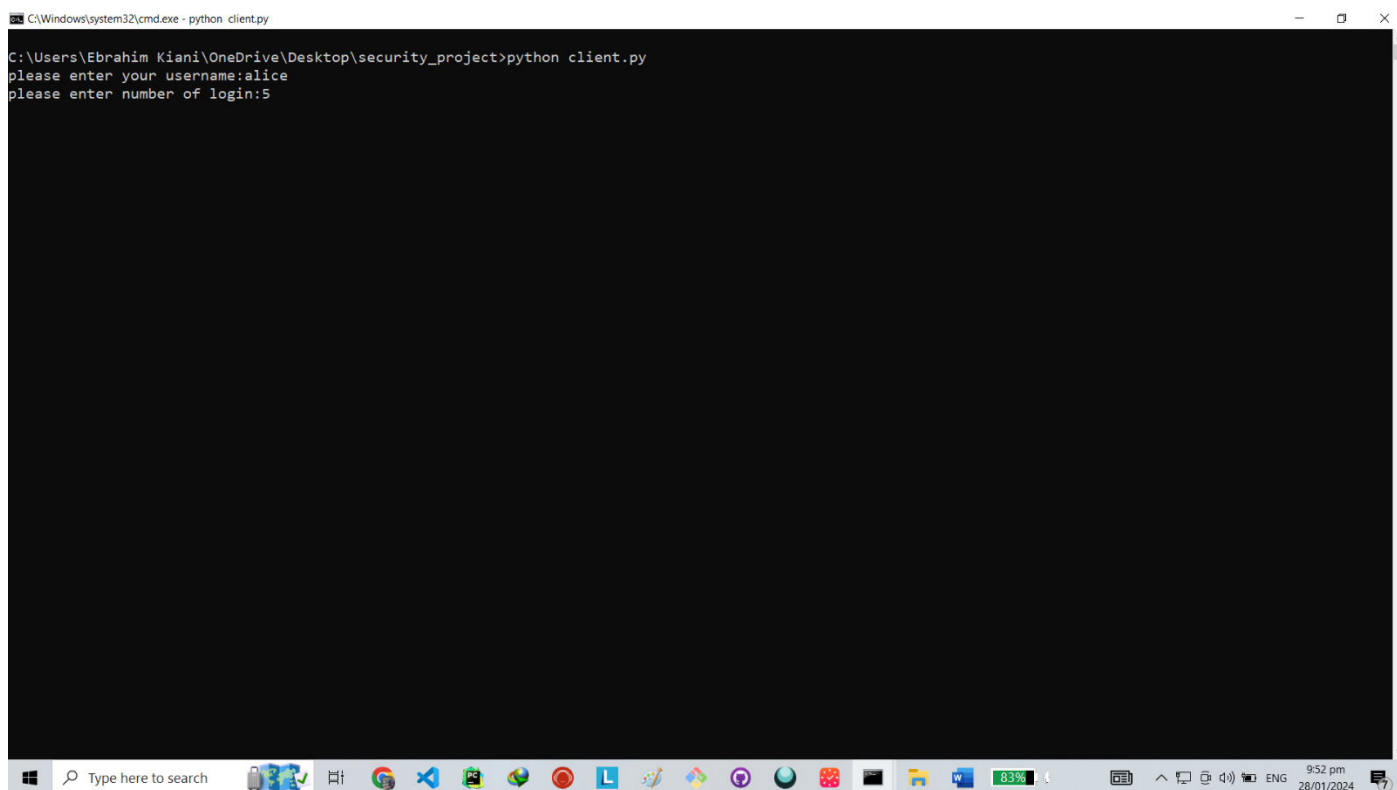
اما در سوی دیگر در سرور مشاهده میکنیم که سرور با موفقیت به کلاینت متصل شده و پیغام اتصال را میدهد و اعلان میکند که در حال شنیدن از کلاینت مد نظر است.

```
C:\Windows\system32\cmd.exe - python server.py

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>python server.py
Server listening on 127.0.0.1:12345

Connected to client:
('127.0.0.1', 52442)
C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project\server.py:16: MovedIn20Warning: The ``declarative_base()`` function is now available as sqlalchemy.orm.declarative_base(). (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  Base = declarative_base()
2024-01-28 21:42:34,955 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-01-28 21:42:34,956 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("Usermodel")
2024-01-28 21:42:34,956 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-01-28 21:42:34,957 INFO sqlalchemy.engine.Engine COMMIT
```

اکنون می‌خواهیم نام alice را برای کاربر انتخاب کنیم و تعداد 5 عدد لاگین را به فایل کلاینت بدهیم. در شکل زیر مشاهده می‌کنید.



```
C:\Windows\system32\cmd.exe - python client.py
C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>python client.py
please enter your username:alice
please enter number of login:5
```

پس از زدن دکمه اینتر در بخش کلاینت، سرور 5 بار رمز کاربر را هش کرده و سپس در سرور ذخیره می‌کند.

سپس کاربر 5 بار لاگین می‌کند و هر بار یک بار کمتر رمز خود را هش می‌کند و برای سرور می‌فرستد. سرور نیز در هر بار رمز کاربر را یک بار هش کرده و سپس بررسی می‌کند که آیا کاربر معتبر است یا خیر. اگر معتبر بود رمز کاربر را در دیتابیس همان چیزی قرار می‌دهد که کاربر برایش فرستاده بود و سپس منتظر لاگین بعدی کاربر می‌شود.

این کار همان 5 بار که داده بودیم تکرار می‌شود تا به خود رمز 1234 کاربر برسیم.

بدین ترتیب زنجیره هش اجرا شده و در هر بار رمز های متفاوتی از طریق سوکت برای سرور ارسال می‌شود.

همانطور که در بخش سرور مشاهده می‌کنیم.

```

Select C:\Windows\system32\cmd.exe

*****"user found"*****

2024-01-28 22:00:21,661 INFO sqlalchemy.engine.Engine UPDATE "Usermodel" SET password=? WHERE "Usermodel".id = ?
2024-01-28 22:00:21,662 INFO sqlalchemy.engine.Engine [cached since 0.01136s ago] ('b0c64e484b2b1c26fec23e2c40ddeda7ac0b4c47f7b70466eed4fd57ac461606', 2)
2024-01-28 22:00:21,663 INFO sqlalchemy.engine.Engine COMMIT
*****Received the username and hashed password form client:
alice 756bc47cb5215dc3329ca7e1f7be33a2dad68990bb94b76d90aa07f4e44a233a
2024-01-28 22:00:21,673 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-01-28 22:00:21,674 INFO sqlalchemy.engine.Engine SELECT "Usermodel".id AS "Usermodel_id", "Usermodel".username AS "Usermodel_username", "Usermodel".password AS "Usermodel_password"
FROM "Usermodel"
WHERE "Usermodel".username = ? AND "Usermodel".password = ?
LIMIT ? OFFSET ?
2024-01-28 22:00:21,674 INFO sqlalchemy.engine.Engine [cached since 0.02488s ago] ('alice', 'b0c64e484b2b1c26fec23e2c40ddeda7ac0b4c47f7b70466eed4fd57ac461606', 1, 0)
*****"user found"*****

2024-01-28 22:00:21,675 INFO sqlalchemy.engine.Engine UPDATE "Usermodel" SET password=? WHERE "Usermodel".id = ?
2024-01-28 22:00:21,675 INFO sqlalchemy.engine.Engine [cached since 0.02413s ago] ('756bc47cb5215dc3329ca7e1f7be33a2dad68990bb94b76d90aa07f4e44a233a', 2)
2024-01-28 22:00:21,676 INFO sqlalchemy.engine.Engine COMMIT
*****Received the username and hashed password form client:
alice 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
2024-01-28 22:00:21,682 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-01-28 22:00:21,682 INFO sqlalchemy.engine.Engine SELECT "Usermodel".id AS "Usermodel_id", "Usermodel".username AS "Usermodel_username", "Usermodel".password AS "Usermodel_password"
FROM "Usermodel"
WHERE "Usermodel".username = ? AND "Usermodel".password = ?
LIMIT ? OFFSET ?
2024-01-28 22:00:21,683 INFO sqlalchemy.engine.Engine [cached since 0.03382s ago] ('alice', '756bc47cb5215dc3329ca7e1f7be33a2dad68990bb94b76d90aa07f4e44a233a', 1, 0)
*****"user found"*****

2024-01-28 22:00:21,684 INFO sqlalchemy.engine.Engine UPDATE "Usermodel" SET password=? WHERE "Usermodel".id = ?
2024-01-28 22:00:21,684 INFO sqlalchemy.engine.Engine [cached since 0.03293s ago] ('03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4', 2)
2024-01-28 22:00:21,685 INFO sqlalchemy.engine.Engine COMMIT
*****Received the username and hashed password form client:
alice 1234
2024-01-28 22:00:21,691 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-01-28 22:00:21,691 INFO sqlalchemy.engine.Engine SELECT "Usermodel".id AS "Usermodel_id", "Usermodel".username AS "Usermodel_username", "Usermodel".password AS "Usermodel_password"
FROM "Usermodel"
WHERE "Usermodel".username = ? AND "Usermodel".password = ?
LIMIT ? OFFSET ?
2024-01-28 22:00:21,692 INFO sqlalchemy.engine.Engine [cached since 0.04301s ago] ('alice', '03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4', 1, 0)
*****"user found"*****

2024-01-28 22:00:21,693 INFO sqlalchemy.engine.Engine UPDATE "Usermodel" SET password=? WHERE "Usermodel".id = ?
2024-01-28 22:00:21,693 INFO sqlalchemy.engine.Engine [cached since 0.0426s ago] ('1234', 2)
2024-01-28 22:00:21,695 INFO sqlalchemy.engine.Engine COMMIT

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>

```

کاربر برسد و ان را ذخیره سازی کند.(در عمل نباید رمز اصلی در دیتا بیس نمایش داده شود و باید هش آخر قرار داده شود ولی در اینجا برای نشان دادن صحت عمل پروژه، در آخر به رمز اصلی رسیده ام)

و اما در فایل کلاینت اکنون مشاهده میکنیم که:

```
C:\Windows\system32\cmd.exe

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>python client.py
please enter your username:alice
please enter number of login:5
Received response from server:
Hello from the server, your number of hash has been recived!

Received response from server:
Hello from the server, your usernme has been recived!

Received logging response from server:
message from the server, your usernme and password have been recived and And your account validation is correct! you are logged in :)

Received logging response from server:
message from the server, your usernme and password have been recived and And your account validation is correct! you are logged in :)

Received logging response from server:
message from the server, your usernme and password have been recived and And your account validation is correct! you are logged in :)

Received logging response from server:
message from the server, your usernme and password have been recived and And your account validation is correct! you are logged in :)

Received logging response from server:
message from the server, your usernme and password have been recived and And your account validation is correct! you are logged in :)

C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project>_
```

ارسال شده است و سرور در پاسخ به کلاینت فهمانده است که او لاگین شده است! در فایل دیتابیس به صورت زیر مشاهده میکنیم که آخرین لاگین کاربر در دیتابیس ثبت شده است.

منطق میانی کد سرور:

DB Browser for SQLite - C:\Users\Ebrahim Kiani\OneDrive\Desktop\security_project\ServerDataBase.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Usermodel

	id	username	password
Filter	Filter	Filter	
1	1	alice	1234

New Record Delete Record

Go to: 1

Edit Database Cell

Mode: Text Import Export Set as NULL

1

Type of data currently in cell: Text / Numeric
1 char(s)

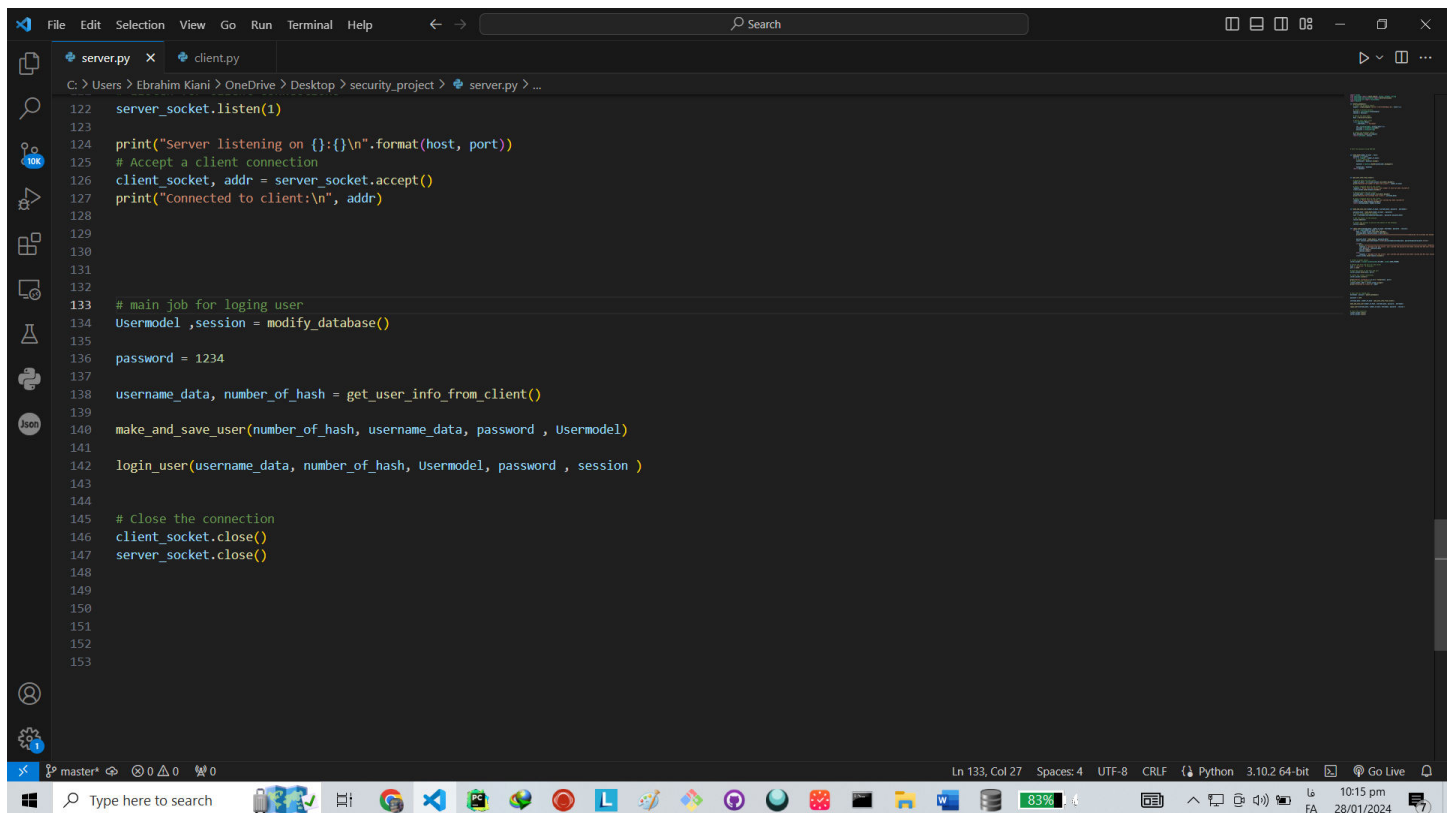
Apply

DB Schema

Name	Type	Schema
Tables (1)		
Usermodel	CREATE TABLE "Usermodel" (
id	INTEGER	"id" INTEGER NOT NULL
usern...	VARCHAR(100)	"username" VARCHAR(100)
pass...	VARCHAR	"password" VARCHAR
Indices (0)		
Views (0)		
Triggers (0)		

SQL Log Plot DB Schema Remote

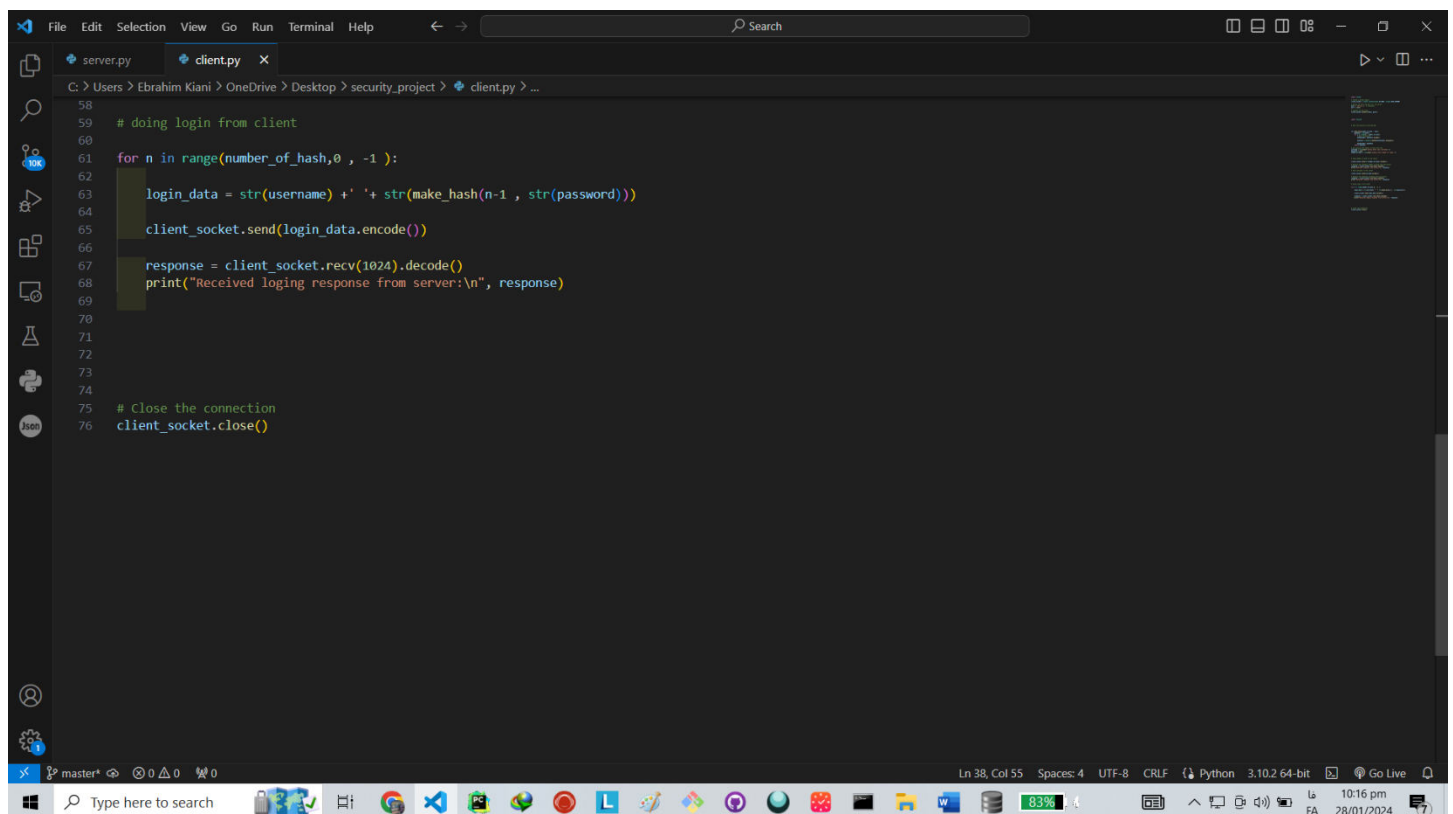
UTF-8



The screenshot shows a Visual Studio Code editor window with a dark theme. The file explorer on the left shows two files: 'server.py' and 'client.py'. The 'server.py' file is open in the editor, showing lines 122 to 153. The code is a Python script for a server. It starts with 'server_socket.listen(1)', followed by a print statement for the server listening on a specific host and port. Then, it enters a loop to accept client connections. For each connection, it prints the client address and performs a login process. The login process involves calling 'get_user_info_from_client()', 'make_and_save_user()', and 'login_user()'. Finally, it closes the client and server sockets. The status bar at the bottom indicates the file is at line 133, column 27, with 4 spaces, UTF-8 encoding, and CRLF line endings. The system tray shows the date as 28/01/2024 and the time as 10:15 pm.

```
122 server_socket.listen(1)
123
124 print("Server listening on {}:{}".format(host, port))
125 # Accept a client connection
126 client_socket, addr = server_socket.accept()
127 print("Connected to client:\n", addr)
128
129
130
131
132
133 # main job for logging user
134 Usermodel ,session = modify_database()
135
136 password = 1234
137
138 username_data, number_of_hash = get_user_info_from_client()
139
140 make_and_save_user(number_of_hash, username_data, password , Usermodel)
141
142 login_user(username_data, number_of_hash, Usermodel, password , session )
143
144
145 # close the connection
146 client_socket.close()
147 server_socket.close()
148
149
150
151
152
153
```

و همینطور منطق پیاده شده در کد کلاینت:



The screenshot shows a Visual Studio Code editor window with a dark theme. The file explorer on the left shows two files: 'server.py' and 'client.py'. The 'client.py' file is open in the editor, showing lines 58 to 76. The code is a Python script for a client. It starts with a comment '# doing login from client'. Then, it enters a loop 'for n in range(number_of_hash, 0, -1):'. Inside the loop, it constructs a 'login_data' string by concatenating the username, a separator, and the result of 'make_hash(n-1, str(password))'. It then sends this data to the server using 'client_socket.send()'. After sending, it receives a response from the server using 'client_socket.recv(1024).decode()' and prints it. Finally, it closes the client socket. The status bar at the bottom indicates the file is at line 38, column 55, with 4 spaces, UTF-8 encoding, and CRLF line endings. The system tray shows the date as 28/01/2024 and the time as 10:16 pm.

```
58
59 # doing login from client
60
61 for n in range(number_of_hash, 0, -1):
62
63     login_data = str(username) + ' ' + str(make_hash(n-1, str(password)))
64
65     client_socket.send(login_data.encode())
66
67     response = client_socket.recv(1024).decode()
68     print("Received logging response from server:\n", response)
69
70
71
72
73
74
75 # Close the connection
76 client_socket.close()
```

در هر دو فایل کامنت ها و تابع های مدنظر را به صورت خوانا قرار داده ام و کافیت به صورت یک نگاه سطحی بررسی کنید تا منطق آن را به طور واضح درک کنید.