

# Assignment 6

IDATT2503 - Security in programming and cryptography  
Fall 2023

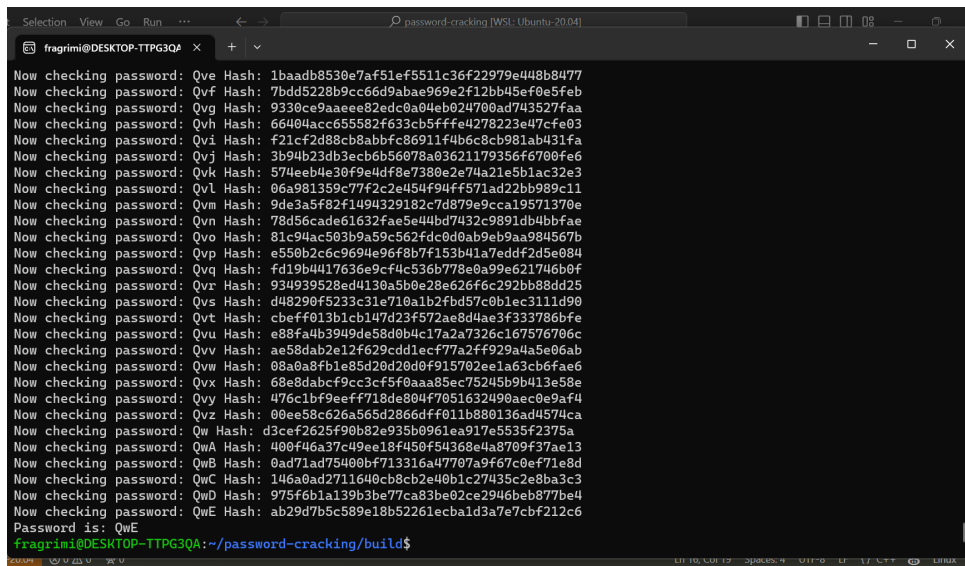
Francesca Grimaldi

## 1 Task 1

For Task 1, I created a C++ program that uses PBKDF2 with SHA1 to crack Ole's password.

As it is possible to see from the code, the `pbkdf2` function takes as input the given salt and number of iterations, as well as an input password that is built bruteforcing characters from a list of possible ones (in this case, uppercase and lowercase letters of the alphabet). The result of the hashing is confronted with the given key: if there is a match, it means that the password is the one we were looking for; otherwise, another attempt needs to be made using a different combination of characters.

In this case, I found that the password was `QwE`.



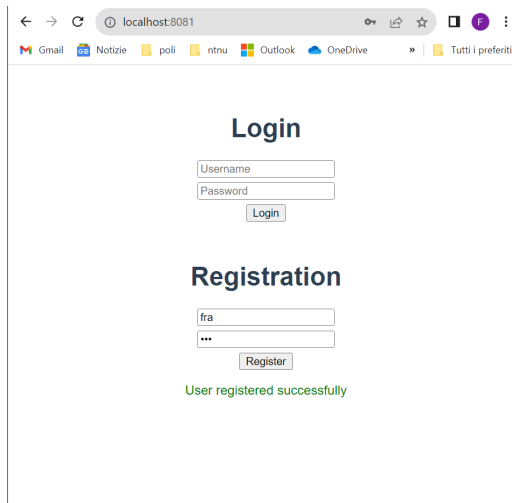
```
Selection View Go Run ... password-cracking [WSL: Ubuntu-20.04]
fragrini@DESKTOP-TTPG3QA: ~
Now checking password: Qve Hash: 1baadb8530e7af51ef5511c36f22979e448b8477
Now checking password: Qvf Hash: 7bdd5228b9cc66d9abae969e2f12bb45ef0e5feb
Now checking password: Qvg Hash: 9330ce9aeeee82edc0a04eb024700ad743527faa
Now checking password: Qvh Hash: 66404acc655582f633cb5fffe4278223e47cfe03
Now checking password: Qvi Hash: f21cf2d88cb8abbfc86911f4b6c8cb981ab431fa
Now checking password: Qvj Hash: 3b94b23db3ecb6b56078a03621179356f670fe6
Now checking password: Qvk Hash: 574eeb4e30f9e4df8e7380e2e74a21e5b1ac32e3
Now checking password: Qvl Hash: 06a981359c77f2c2e454f94ff571ad22bb989c11
Now checking password: Qvm Hash: 9de3a5f82f1494329182c7d879e9cca19571370e
Now checking password: Qvn Hash: 78d56cade61632fae5e44bd7432c9891db4bbfae
Now checking password: Qvo Hash: 81c94ac503b9a59c562fcd0d0ab9eb9aa984567b
Now checking password: Qvp Hash: e550b2c6c9694e96f8b7f153b41a7eddf2d5e084
Now checking password: Qvq Hash: fd19b4417636e9cf4c536b778e0a99e621746b0f
Now checking password: Qvr Hash: 934939528ed4130a5b0e28e626f6c292bb88dd25
Now checking password: Qvs Hash: d48290f5233c31e710a1b2fbd57c0b1ec3111d90
Now checking password: Qvt Hash: cbeff013b1cb147d23f572ae8d4ae3f33786bfe
Now checking password: Qvu Hash: e88fa4b3949de58d0b4c17a2a7326c167576706c
Now checking password: Qvv Hash: ae58dab2e12f629cdd1ecf77a2ff929a4a5e06ab
Now checking password: Qvw Hash: 08a0a8fb1e85d20d20d0f915702ee1a63cb6fae6
Now checking password: Qvx Hash: 68e8dabc9cc3cf5f0aaa85ec75245b9b413e58e
Now checking password: Qvy Hash: 476c1bf9eeff718de804f7051632490aec0e9af4
Now checking password: Qvz Hash: 00ee58c626a565d2866dff011b880136ad4574ca
Now checking password: Qw Hash: d3cef2625f90b82e935b0961ea917e5535f2375a
Now checking password: QwA Hash: 400f46a37c49ee18f450f54368e4a8709f37ae13
Now checking password: QwB Hash: 0ad71ad75400bf713316a47707a9f67c0ef71e8d
Now checking password: QwC Hash: 146a0ad2711640cb8cb2e40b1c27435c2e8ba3c3
Now checking password: QwD Hash: 975f6b1a139b3be77ca83be02ce2946beb877be4
Now checking password: QwE Hash: ab29d7b5c589e18b52261ecba1d3a7e7cbf212c6
Password is: QwE
fragrini@DESKTOP-TTPG3QA: ~/password-cracking/build$
```

Figure 1: Found password

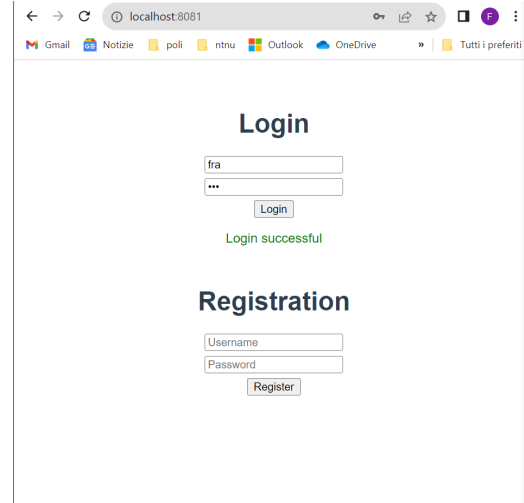
## 2 Task 2

For Task 2, I created the web client and server using Vue. PBKDF2 is used for hashing the password and for the authentication. The homepage presents the Login and Registration components, in which the input fields are validated and the password is hashed (client side) using the JavaScript `crypto-js` library. On the server side, Node.js `Crypto` library is used.

Some screenshots of different scenarios are the following:

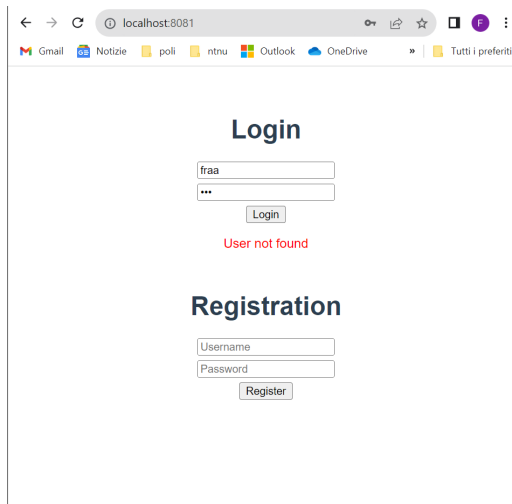


(a) Successful registration

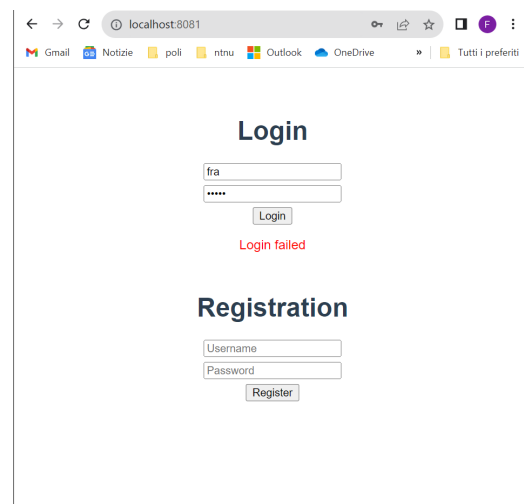


(b) Successful login

Figure 2: Successful login and registration



(a) Attempt to login with wrong username



(b) Attempt to login with wrong password

Figure 3: Wrong credentials

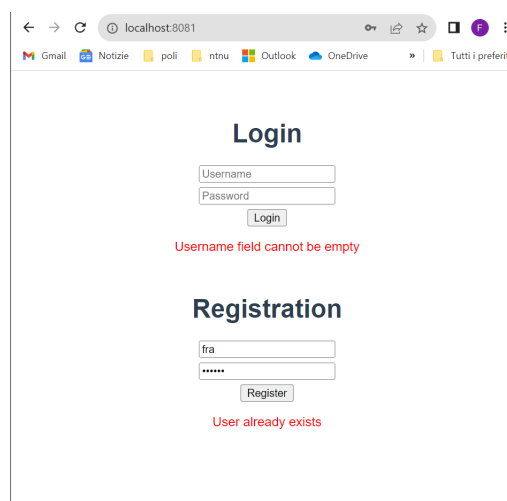


Figure 4: Attempt to login with empty fields and to register with already existing username