Sigurnost računala i podataka - Vježbe 5

Cilj je vidit razliku imežu sporih i brzih kriptografskih funkcija.

Bilo potrebno instalirat potrebne pakete i zalijepit kod te pokrenit ga. Rezultat je da je vrijeem sporih hash sekunda svejedno u milisekundama tj malo je. Iako pari malo vrijeme, kad se pomnozi s npr 10000, razlika je evidentna tako da zbog ekonomicnosti odvrati napadaca od pokusaja.

```
+----------+---------------------+
| Function | Avg. Time (100 runs) |
+----------+---------------------+
| AES      |       0.002364      |
+----------+---------------------+


+----------+---------------------+
| Function | Avg. Time (100 runs) |
+----------+---------------------+
| HASH_MD5 |       0.00028       |
| AES      |       0.002364      |
+----------+---------------------+


+-------------+---------------------+
| Function    | Avg. Time (100 runs) |
+-------------+---------------------+
| HASH_SHA256 |       0.000183      |
| HASH_MD5    |       0.00028       |
| AES         |       0.002364      |
+-------------+---------------------+
```

```
Zaključci se ne mogu donosit na temelju jednog pokretanja već prosjeka npr 100 pokretanja.
Isto tako nije pozeljno koristit for petlju jer je lako za napadaca da je probije tj samo je izbrise.
linux_crypt_6 100 puta sporiji od hash_sha256

linux_crypt 100k 1000 puta sporiji

scrypt_n_2_18 posjek 1.66sek


import sqlite3
from sqlite3 import Error
from passlib.hash import argon2
```

```
import getpass

import sys
from InquirerPy import inquirer
from InquirerPy.separator import Separator


def verify_password(password: str, hashed_password: str) -> bool:
    # Verify that the password matches the hashed password
    return argon2.verify(password, hashed_password)

def get_user(username):
    try:
        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username = ?", (username,))
        user = cursor.fetchone()
        conn.close()
        return user
    except Error:
        return None


def register_user(username: str, password: str):
    # Hash the password using Argon2
    hashed_password = argon2.hash(password)

    # Connect to the database
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()

    # Create the table if it doesn't exist
    cursor.execute(
        "CREATE TABLE IF NOT EXISTS users (username TEXT PRIMARY KEY UNIQUE, password TEXT)"
    )

    try:
        # Insert the new user into the table
        cursor.execute("INSERT INTO users VALUES (?, ?)", (username, hashed_password))

        # Commit the changes and close the connection
        conn.commit()
    except Error as err:
        print(err)
    conn.close()


def do_register_user():
    username = input("Enter your username: ")

    # Check if username taken
    user = get_user(username)
    if user:
        print(f'Username "{username}" not available. Please select a different name.')
        return

    password = getpass.getpass("Enter your password: ")
    register_user(username, password)
    print(f'User "{username}" successfully created.')
```

```
def do_sign_in_user():
    username = input("Enter your username: ")
    password = getpass.getpass("Enter your password: ")
    user = get_user(username)

    if user is None:
        print("Invalid username or password.")
        return

    password_correct = verify_password(password=password, hashed_password=user[-1])

    if not password_correct:
        print("Invalid username or password.")
        return
    print(f'Welcome "{username}".')


if __name__ == "__main__":
    REGISTER_USER = "Register a new user"
    SIGN_IN_USER = "Login"
    EXIT = "Exit"

    while True:
        selected_action = inquirer.select(
            message="Select an action:",
            choices=[Separator(), REGISTER_USER, SIGN_IN_USER, EXIT],
        ).execute()

        if selected_action == REGISTER_USER:
            do_register_user()
        elif selected_action == SIGN_IN_USER:
            do_sign_in_user()
        elif selected_action == EXIT:
            sys.exit(0)
```