Peta laboratorijska vjezba

U petoj laboratorijskoj vježbi simulirali smo autentifikaciju korisnika putem password-a.

U našem programu implementiran je izbornik s tri opcije: registriranje novog korisnika, prijava postojećeg korisnika i exit.

Prilikom registracije korisnika potrebno je provjeriti postoji li korisnik s već postojećim korisničkim imenom. Svaki korisnik mora imati jedinstveno korisničko ime. Kada bi novi korisnik zaželio imati isti password kao neki drugi postojeći korisnik, to ne bi ugrozilo sigurnost sustava jer koristimo password salting, odnosno prilikom dodijeljivanja password-a dodijelimo i random salt vrijednost (koja ne mora biti tajna). Tako bi dva korisnika s identičnim password-ima imali različite Hash(password + salt) vrijednosti, čime u bazi podataka ne bismo spremali dvije identične hash vrijednosti. Kada ne bismo koristili salt, ta dva korisnika bi imala identične hash-eve pa bi napadač znao da imaju iste passworde (to može zaključiti zbog collision resistance svojstva).

Prilikom prijave postojećeg korisnika, sustav traži unos korisničkog imena i password-a, a tek onda provjerava postoji li već korisnik s danim username-om. Kada bi sustav prvo provjerio postoji li dano ime, a tek onda zatražio unos passworda, napadač bi mogao sa sigurnošću znati koji username-ovi postoje u bazi, a koji ne.

Odgovorite:

1. **Koliko korisnika je registrirano u bazu podataka?** 3
2. **Usporedite hash vrijednosti zaporki korisnika `jdoe` i `jean_doe`. Što možete zaključiti?** Različite su hash vrijednosti, a isti password-i, znači da se koristimo salt metodom.

Odgovorite:

1. **Zašto pri provjeri unesene zaporke `argon2` funkcija treba oboje, zaporku i njenu `hash` vrijednost?** Kako bi mogao usporediti unesenu zaporku sa

spremljenim hashom ispravne zaporke.

2. **Koji još važan element treba `argon2` za ispravnu provjeru unesene zaporke?**
   Salt vrijednost.

Odgovorite:

1. **Zašto u funkciji `do_sign_in_user()` tražimo od korisnika da uvijek unese oboje, `username` i `password`, čak iako `username` potencijalno nije ispravan?**

   Prilikom prijave postojećeg korisnika, sustav traži unos korisničkog imena i password-a, a tek onda provjerava postoji li već korisnik s danim username-om. Kada bi sustav prvo provjerio postoji li dano ime, a tek onda zatražio unos passworda, napadač bi mogao sa sigurnošću znati koji username-ovi postoje u bazi, a koji ne.

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

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

import getpass


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 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 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}".')


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

```
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)
```