Actividad práctica: Almacenamiento y autenticación de contraseñas

Objetivos de la actividad

- En este laboratorio, los estudiantes aprenderán a almacenar contraseñas de forma segura utilizando el algoritmo de hashing Argon2 y la técnica de sal y pimienta. También aprenderán a autenticar usuarios de forma segura y a proteger las contraseñas almacenadas en la base de datos.

Requisitos previos:

- Conocimiento básico de Python y Flask.
- Tener instalado Python 3 y Flask.
- Tener instalada la librería argon2-cffi para Python.

Configuración de la base de datos:

1. Descargar e instalar SQLite en el sistema.
2. Abrir la línea de comandos y navegar hasta el directorio donde se encuentra el archivo de la aplicación Flask.
3. Ejecutar el siguiente comando para abrir una consola de SQLite:

```
> sqlite3 password.db
```

4. Crear la tabla de usuarios ejecutando el siguiente comando:

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    salt VARCHAR(255) NOT NULL
);
```

5. Salir de la consola de SQLite ejecutando el siguiente comando:

```
> .exit
```

Almacenamiento y autenticación de contraseñas: 

6. Crear una aplicación Flask con las rutas de autenticación y almacenamiento de contraseñas.
7. Importar la librería argon2 y definir una constante para el número de rondas de hashing a utilizar.
8. Crear una función para generar una sal aleatoria para cada contraseña y una función para generar un hash seguro utilizando la librería argon2 y la técnica de sal y pimienta.
9. Crear una ruta para el registro de usuarios que almacene la información del usuario en la base de datos utilizando el hash generado en el paso anterior.

10. Crear una ruta para la autenticación de usuarios que verifique las credenciales del usuario utilizando el hash almacenado en la base de datos.
11. Añadir verificaciones de seguridad adicionales a la ruta de registro y autenticación, como la comprobación de que el nombre de usuario y la contraseña son válidos y la prevención de ataques de fuerza bruta.
12. Probar la aplicación registrando y autenticando usuarios y verificando que las contraseñas se almacenan de forma segura en la base de datos.
13. Mejorar la seguridad de la aplicación utilizando técnicas adicionales, como la limitación del número de intentos de inicio de sesión fallidos y la aplicación de una política de contraseñas seguras.

Código de referencia:

```python
from flask import Flask, request, redirect, url_for, render_template
from argon2 import PasswordHasher

app = Flask(__name__)
ph = PasswordHasher(hash_len=32, salt_len=16, time_cost=2, memory_cost=102400)

# Funciones de hashing y salado
def generate_salt():
    return os.urandom(16)

def hash_password(password, salt):
    pepper = b'MySuperSecretPepper'
    password_peppered = password.encode() + pepper
    return ph.hash(password_peppered, salt)

# Rutas
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        salt = generate_salt()
        password_hash = hash_password(password, salt)
        db.execute('INSERT INTO users (username, password_hash, salt) VALUES (?, ?, ?)',
                   (username, password_hash, salt))
        db.commit()
        return redirect(url_for('login'))
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = db.execute('SELECT * FROM users WHERE username = ?', (username,)).fetchone()
        if user and ph.verify(user['password_hash'], password.encode() + b'MySuperSecretPepper'):
            session['user_id'] = user['id']
            return redirect(url_for('dashboard'))
        else:
            flash('Incorrect username or password')
    return render_template('login.html')

@app.route('/dashboard')
def dashboard():
    if 'user_id' in session:
        user = db.execute('SELECT * FROM users WHERE id = ?', (session['user_id'],)).fetchone()
        return render_template('dashboard.html', username=user['username'])
    return redirect(url_for('login'))

# Configuración de la base de datos
import os
import sqlite3

db = sqlite3.connect('password.db')
db.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE, password_hash TEXT, salt TEXT)')
db.commit()

if __name__ == '__main__':
    app.secret_key = os.urandom(16)
    app.run(debug=True)
```

Este código es solo un ejemplo y no está optimizado para un entorno de producción. Se deben implementar medidas de seguridad adicionales antes de implementar una aplicación en un entorno de producción real.

Explicación de la mitigación

En este ejemplo se utilizan técnicas de almacenamiento seguro de contraseñas mediante el uso de Argon2, sal y pimienta. Argon2 es un algoritmo de hashing que utiliza una función de mezcla de memoria intensiva para hacer más difícil la fuerza bruta y otros ataques de diccionario. Además, se genera una salt aleatoria única para cada usuario, lo que hace que incluso dos usuarios con la misma contraseña

tengan hashes diferentes. La pimienta es un valor secreto que se agrega a la contraseña antes de hacer el hash, lo que hace que sea aún más difícil para un atacante obtener las contraseñas incluso si obtienen acceso a la base de datos de contraseñas.

En cuanto a la implementación en Flask, se utiliza el paquete "argon2" para manejar el hashing de contraseñas y se almacena la salt y la contraseña hash en una base de datos SQLite. Además, se utiliza la función session de Flask para almacenar temporalmente la información de inicio de sesión del usuario en el servidor web.

Preguntas de evaluación de la actividad

Asegúrate de saber responder las siguientes preguntas e incluye tus respuestas en el reporte:

1. **¿Por qué es importante utilizar técnicas de almacenamiento seguro de contraseñas en las aplicaciones web?**
2. **¿Qué es la "salt" en el contexto del hashing de contraseñas y por qué es importante?**
3. **¿Qué es la "pimienta" en el contexto del hashing de contraseñas y por qué es importante?**
4. **¿Qué es Argon2 y cómo se utiliza para almacenar contraseñas de forma segura?**
5. **¿Cuál es la diferencia entre un ataque de fuerza bruta y un ataque de diccionario, y cómo se pueden prevenir en la autenticación de contraseñas en una aplicación web?**

✋ Fin de la actividad. Tómate un descanso, prepara el reporte y envíalo como tarea.