LU06k - Passwortsicherheit

Sicherheit ist ein entscheidender Aspekt in jeder Webanwendung. Ein wichtiges Element der Sicherheit ist die sichere Speicherung von Passwörtern. In dieser Lektion konzentrieren wir uns auf das Hashing von Passwörtern.

Warum Passwort-Hashing?

In einer sicheren Anwendung sollten Passwörter niemals im Klartext gespeichert werden. Anstelle dessen sollten sie gehasht werden, und der Hash-Wert sollte in der Datenbank gespeichert werden. Dies macht es für Angreifer schwierig, das ursprüngliche Passwort wiederherzustellen, selbst wenn sie Zugriff auf die Datenbank erhalten.

Verwendung von bcrypt

Python bietet verschiedene Bibliotheken für das Passwort-Hashing, eine der beliebtesten ist `bcrypt`. Sie können bcrypt mit pip installieren:

```
pip install bcrypt
```

oder im `requirements.txt` ergänzen:

requirements.txt

```
...
bcrypt==4.0.1
```

Hashing in der DAO-Klasse

Nach der Installation können Sie bcrypt in Ihrer DAO-Klasse verwenden, um das Passwort zu hashen:

```
import bcrypt

class UserDao:
    # ...
    def add_user(self, user):
        hashed_pw = bcrypt.hashpw(user.password.encode('utf-8'),
bcrypt.gensalt())
        self.cursor.execute("INSERT INTO users (username, email, password)
VALUES (?, ?, ?)", (user.username, user.email, hashed_pw))
        self.conn.commit()
    # ...
```

Authentifizierung

Bei der Authentifizierung wird das vom Benutzer eingegebene Passwort ebenfalls gehasht und mit dem in der Datenbank gespeicherten Hash verglichen:

```
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    user = user_dao.get_user_by_username(data['username'])
    if user and bcrypt.checkpw(data['password'].encode('utf-8'),
user.password):
        login_user(user)
        return jsonify({'success': True}), 200
    return jsonify({'error': 'Invalid username or password'}), 401
```

Durch die Verwendung von Passwort-Hashing erhöhen Sie die Sicherheit Ihrer Anwendung erheblich.

M323-LU06