SecCode Analyzer

Analysis Results

Analysis of Security Issues

1. **SQL Injection Vulnerability

**

- **Severity Level:** Critical

- **Description:** The code directly inserts user inputs (username and password) into SQL queries without sanitization, making it vulnerable to SQL injection attacks.

- **Suggested Improvement:** Use parameterized queries to prevent SQL injection attacks.

- **Example Improvement:**

```python

def add_user(username, password):

cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, password))

conn.commit()

## def authenticate(username, password)

cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?", (username, password))

user = cursor.fetchone()

if user:

print("Autenticado com sucesso!")

else:

print("Falha na autenticação!")

```

2. **Plain Text Password Storage

**

- **Severity Level:** High

- **Description:** Storing passwords in plain text in the database is a significant security risk as it exposes user credentials if the database is compromised.

- **Suggested Improvement:** Hash the passwords before storing them in the database.

- **Example Improvement:**

```python

import hashlib

## def add_user(username, password)

hashed_password = hashlib.sha256(password.encode()).hexdigest()

cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, hashed_password))

conn.commit()

## def authenticate(username, password)

hashed_password = hashlib.sha256(password.encode()).hexdigest()

cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?", (username, hashed_password))

user = cursor.fetchone()

if user:

print("Autenticado com sucesso!")

else:

print("Falha na autenticação!")

```

3. **Input Sanitization

**

- **Severity Level:** Medium

- **Description:** The code does not perform input validation or sanitization, which can lead to unexpected behavior or security vulnerabilities.

- **Suggested Improvement:** Implement input validation and sanitization to prevent malicious input.

- **Example Improvement:** Implement basic input validation for username and password fields.

Summary

The code snippet provided has critical security vulnerabilities such as SQL injection and high-risk issues like storing passwords in plain text. By implementing the suggested improvements, such as using parameterized queries, hashing passwords, and input validation, the code can significantly enhance its security posture. Remember to always follow secure coding practices to protect sensitive data and prevent security breaches.