VAmPI API - Code Review Report

General Overview

This is a code review of VAmPI, which is a vulnerable REST API written in Python using Flask. The idea behind it is to show common security issues in APIs, especially the kind listed in OWASP's top 10. I went through the code and tried some of the endpoints to see how they behave.

Where I Found Issues

1. SQL Injection

- File: routes/users.py, endpoint /api/users/login

- Problem: The login function uses raw SQL queries with user input directly in the query string.

cursor.execute(f"SELECT \* FROM users WHERE username = '{username}' AND password = '{password}'")

- Why it’s bad: Anyone can input something like admin'-- and log in without a real password.

- Fix: Should use parameterized queries instead.

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

2. Plaintext Passwords and No Real Authentication

- File: Also in routes/users.py

- Problem: Passwords are saved as-is in the database. No sessions, no tokens.

- Risk: If someone gets the DB, they get everyone's password.

- Fix: Should hash passwords with bcrypt or similar, and use JWTs or Flask sessions for login.

3. Returning Too Much Data

- File: routes/users.py, endpoint /api/users/all

- Problem: It returns all user info including passwords and emails.

- Fix: Just return the needed parts, like this:

return jsonify([{"id": u[0], "username": u[1]} for u in users])

4. Unsafe File Uploads

- File: routes/uploads.py

- Problem: Uploaded files are saved without checking file type or renaming.

- Fix: Check file extensions and rename files before saving. Only allow specific types.

5. Debug Mode is On

- File: vampi.py

- Problem: Flask is running in debug mode which should only be used in development.

- Fix: Turn it off in production:

app.run(debug=False)

6. No Rate Limiting

- Problem: All endpoints can be spammed.

- Fix: Use Flask-Limiter or some other method to restrict requests per IP.

Final Report

The project is meant to be insecure, and it does a good job showing a bunch of typical API problems. If this was a real-world app, the issues above would be a big deal. Even though it’s just for practice, it’s still useful to walk through and apply fixes like the ones listed here.

References I Used

- OWASP API Top 10: https://owasp.org/www-project-api-security/

- Flask Security Docs: https://flask.palletsprojects.com/en/latest/security/

- Preventing SQL Injection in Python: https://realpython.com/prevent-python-sql-injection/