

## Database Hardening

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:

psql (18.1)
WARNING: Console code page (437) differs from Windows code page (1252)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \c gradscrape
You are now connected to database "gradscrape" as user "postgres".
gradscrape=# CREATE USER app_worker WITH PASSWORD 'gradDataPass';
CREATE ROLE
gradscrape=# GRANT CONNECT ON DATABASE gradscrape TO app_worker;
GRANT
gradscrape=# GRANT USAGE ON SCHEMA public TO app_worker;
GRANT
gradscrape=# GRANT SELECT, INSERT, UPDATE ON TABLE applicantdata TO app_worker;
GRANT
gradscrape=# GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO app_worker;
GRANT
gradscrape=# |
```

For my Least-Privilege Database user, I granted them the permissions of CONNECT, USAGE, SELECT, INSERT, and UPDATE. I gave the user CONNECT, USAGE, and SELECT permissions, so they are able to see information posted on my website. CONNECT, USAGE, and SELECT allow the user to connect to the database, look up the table associated with my website, and view it respectively. I gave the user INSERT and UPDATE permissions so that they are able to use the buttons on my webpage to scrape new data from the GradCafe survey and add it to the database, as well as update the answers to the queries on the Analysis page. These permissions are enough to allow the user to use my website as intended, without giving them any extra permissions.

## Python Dependency Graph

Based on the dependency.svg file, the key dependencies in my web app are dotenv, psycopg.sql, src.config, psycopg, flask, bs4, src.query\_data, src.load\_data, and web\_app.run. Dotenv is a security dependency that load my environment variables into the application from the .env file. Another important security dependency is psycopg.sql, this provides tools for secure SQL queries, and helps defend against SQL Injections. Config acts as a main hub for my application settings, as it distributes authorized configuration details to the rest of my system. Flask is a key dependency for my web framework, and handles routing and requests. The key dependencies for my web scrape are bs4, which parses HTML content from The GradCafe, src.query\_data, which gets information from the database, and src.load\_data, which serves as an in-between for the web scraper and the database. My last key dependency is web\_app.run,

which is the entry point to my web app, this is key as it initializes my flask app and ensures all dependencies are correctly linked before allowing users onto the server.

```
(.venv) PS C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5> snyk-win test --file=requirements.txt

Testing C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5...

Tested 32 dependencies for known issues, found 1 issue, 1 vulnerable path.

Issues to fix by upgrading dependencies:

Upgrade flask@3.1.2 to flask@3.1.3 to fix
X Use of Cache Containing Sensitive Information (new) [Low Severity][https://security.snyk.io/vuln/SNYK-PYTHON-FLASK-15322678] in flask@3.1.2
  introduced by flask@3.1.2

Organization:    shaylahirji
Package manager: pip
Target file:     requirements.txt
Project name:    module_5
Open source:     no
Project path:    C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5
Licenses:       enabled

Tip: Detected multiple supported manifests (2), use --all-projects to scan all of them at once.

(.venv) PS C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5>
```

During the Snyk dependency scan, one Low Severity vulnerability was identified. My requirements.txt included flask==3.1.2, instead of flask==3.1.3, which uses cache containing sensitive information. To fix this vulnerability, I updated my requirements.txt to show flask==3.1.3 instead.

```
(.venv) PS C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5> snyk-win code test

Testing C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5 ...

Open Issues

X [MEDIUM] Path Traversal
Finding ID: 3f98da82-71ed-4f9a-b161-e66c11be9e65
Path: src/web_scrape/llm_hosting/app.py, line 356
Info: Unsantitized input from a command line argument flows into open, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

X [MEDIUM] Path Traversal
Finding ID: a793d218-64ee-4b2d-8aaf-224681c7c97d
Path: src/web_scrape/llm_hosting/app.py, line 363
Info: Unsantitized input from a command line argument flows into open, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write arbitrary files.

X [MEDIUM] Path Traversal
Finding ID: be716d44-ae87-43e9-b6a3-39507bf1b259
Path: src/web_scrape/llm_hosting/app.py, line 374
Info: Unsantitized input from a command line argument flows into json.dump, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attack
er to write arbitrary files.

Test Summary

Organization:    shaylahirji
Test type:       Static code analysis
Project path:    C:\Users\shayl\JHU\2025\Classes\Test\Modern Concepts in Software\jhu_software_concepts\Module_5_Complete\module_5

Total issues:    3
Ignored issues:  0 [ 0 HIGH  0 MEDIUM  0 LOW ]
Open issues:    3 [ 0 HIGH  3 MEDIUM  0 LOW ]
```

During the Snyk Code Test, 3 Medium Severity issues were identified. All three are identified to be within the llm\_hosting/app.py file, and are all Path Traversal issues. The errors state that an unsantitized command line input is used as a path, which opens up the risk of attackers being able to read or write arbitrary files.