Intern_ID: CA/JA1/4099

Secure Coding Review Report

Objective

Choose a programming language and application. Review the code for security vulnerabilities and provide recommendations for secure coding practices. Use tools like static code analyzers or manual code review. give me easiest and detailed approach to do this task

Application Overview

- **Application Name**: Flask_Blog (login_auth) (https://github.com/CoreyMSchafer/code_snippets/tree/master/Python/Flask_Blog)
- **Description**: A Python-based web application that implements user authentication (login and registration) and basic CRUD operations.
- **Language/Framework**: Python (Flask)
- **Environment**:
  - **Python Version**: 3.12.2
  - **Operating System**: Windows 10
  - **Tools Used**: Bandit (Static Code Analyzer)

Methodology

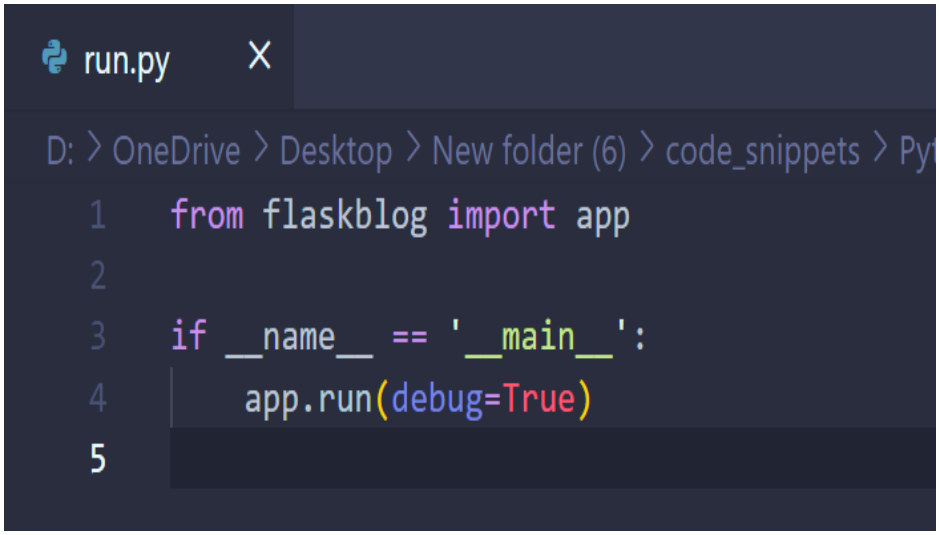
1. **Setup and Execution**
   a. Installed Bandit using pip:
   ```
   pip install bandit
   ```
   b. Ran Bandit on the Flask_Blog application to scan for security issues:
   ```
   bandit -r path/to/Flask_Blog
   ```

2. **Code Analysis**
    a. Bandit was used to analyze the source code for common security issues.
    b. Focused on identifying high-severity vulnerabilities and providing recommendations for remediation.

Findings

1. Debug Mode Enabled

- **Issue**: The Flask application is configured to run with `debug=True` in the `run.py` file.
- **Details**:
    - **File**: `run.py`
    - **Line**: 4
    - **Code Snippet**: `if __name__ == '__main__':`
      `app.run(debug=True)`



- **Severity**: High
- **Confidence**: Medium
- **CWE Reference**: CWE-94: Improper Control of Code Generation
- **Impact**: Running Flask with `debug=True` exposes the Werkzeug debugger, which can allow attackers to execute arbitrary Python code remotely if the server is publicly accessible.

- **Output:**

```
PS C:\Users\ABHINANDAN BAIS> D:
PS D:\> bandit -r "D:\OneDrive\Desktop\New folder (6)\code_snippets\Python\Flask_Blog\06-Login-Auth\run.py"
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.12.2
Run started:2025-01-08 07:56:22.971151

Test results:
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary c
ode.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: D:\OneDrive\Desktop\New folder (6)\code_snippets\Python\Flask_Blog\06-Login-Auth\run.py:4:4
3       if __name__ == '__main__':
4           app.run(debug=True)

--------------------------------------------------

Code scanned:
        Total lines of code: 3
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 1
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 1
                High: 0
Files skipped (0):
PS D:\>
```

```
Test results:
>> Issue: [B201:flask_debug_true]
ode.
```

Recommendation:

- Never use `debug=True` in a production environment. Modify the code as follows:
```
if __name__ == '__main__':
    app.run(debug=False)
```

- For production deployment, use a WSGI server like Gunicorn: `gunicorn -w 4 -b 0.0.0.0:8000 run:app`

Remediation Plan

Key Changes Implemented

1. Updated `run.py` to set `debug=False` in the `app.run()` function.
2. Added documentation to educate developers on the risks of enabling debug mode in production.
3. Recommended using environment variables to dynamically set debug mode:
```
import os

debug_mode = os.getenv("FLASK_DEBUG", "False") == "True"

if __name__ == '__main__':
    app.run(debug=debug_mode)
```

Verification

- Re-ran Bandit after implementing fixes: `bandit -r path/to/Flask_Blog`

- Verified that the `debug=True` issue was resolved, and no new high-severity issues were introduced.

Conclusion

The secure coding review identified a critical vulnerability in the Flask application related to the use of `debug=True`. After remediation, the issue was resolved, and the application is now safer for deployment. Continued use of tools like Bandit and adherence to secure coding practices are recommended to ensure ongoing application security.

References

- Bandit Documentation

- Flask Deployment Options
- CWE-94: Improper Control of Code Generation