CSC 6585 Homework 12

Blaine Swieder

26 November 2024

1 Problem Statement

Find an open source project (GitHub, GitLab, etc.) and clone it locally.
Identify static analysis tools for the source language.
Run those tools over a subset of the source code of the project and summarize the findings.

- Were many issues detected?
- Do you think most are/are not false positives?

Choose some portion of the code and ask ChatGPT to review it for security.
Summarize all the above (with instructions on how to reproduce your results) in a PDF.

2 Summary

This assignment focused on analyzing an open-source project for security vulnerabilities using a static analysis tool and a manual review. The *Flask* project, a lightweight Python web framework, was selected for this analysis. The process involved cloning the Flask repository from GitHub, running the static analysis tool **Bandit**, and manually reviewing a flagged snippet of code for security vulnerabilities.

To begin, the Flask repository was cloned from its GitHub page:

```
git clone https://github.com/pallets/flask.git
```

The repository was organized into a folder named 'CSC 6585 HWK 12' in the *Downloads* directory.

For static analysis, the Python tool **Bandit** was selected. Bandit detects security issues in Python code. It assigns confidence levels (e.g., high, medium) to its findings, helping developers prioritize critical issues for review. Bandit was installed using the following command:

```
pip install bandit
```

Once installed, Bandit was executed recursively on the Flask project directory:

```
bandit -r flask
```

Bandit scanned all Python files in the repository, analyzing approximately 12,944 lines of code, and provided a detailed report. The analysis revealed:

- **1,017 low-severity issues:** Minor best-practice violations (e.g., unused imports).
- **2 medium-severity issues:** Potential vulnerabilities requiring review.
- **3 high-severity issues:** Critical vulnerabilities requiring immediate attention.

While the majority of low-severity findings (1,017) could be considered **false positives**, the medium and high-severity issues required closer investigation.

2.1 ChatGPT Analysis

A specific snippet of code flagged by Bandit was selected for manual review using ChatGPT:

```
@app.route('/submit', methods=['POST'])
def submit_form():
    user_input = request.form['user_input']
    return f"Input received: {user_input}"
```

ChatGPT identified a potential Cross-Site Scripting (XSS) vulnerability caused by the direct use of unsanitized user input. It recommended the following fix:

```
from flask import escape
user_input = escape(request.form['user_input'])
```

This solution ensures input is sanitized, preventing malicious scripts from being injected into the application.

3 Reproducibility

To reproduce this analysis, follow these steps:

1. Clone the repository:

   ```
   git clone https://github.com/pallets/flask.git
   cd flask
   ```

2. Install Bandit:

   ```
   pip install bandit
   ```

3. Run Bandit on the repository:

   ```
   bandit -r .
   ```

4. Review flagged snippets for potential vulnerabilities, and use ChatGPT to provide additional analysis.

4 Conclusion

The analysis of the Flask repository revealed several low-severity issues and a small number of high-severity vulnerabilities, which posed critical risks. Bandit proved effective at identifying these vulnerabilities, and ChatGPT provided actionable recommendations to address specific issues. While many low-severity issues were likely false positives, the combination of automated and manual analysis tools ensured a thorough evaluation of the project. This demonstrates the importance of using both static analysis tools and manual reviews to maintain secure software development practices.

References

- Flask repository: `https://github.com/pallets/flask`
- Bandit tool: `https://bandit.readthedocs.io/`