# APACHE COMMONS-IMAGING LIBRARY

## Dependability Analysis

https://github.com/MerendaFrancoN/commons-imaging-dependability/

Santiago Morales Henao
Julian Orcinoli
Franco Nicolás Merenda

January
2024

# CONTENT


# PROJECT GOALS

Conduct an analysis of software dependability in the context of the Apache Commons Imaging project with a multifaceted approach; leveraging software analytics, testing tools, and vulnerability assessment techniques to evaluate and enhance the dependability and code quality of the project

# PROJECT MANAGEMENT

## JIRA, GITHUB AND OVERLEAF

### PROJECT MANAGEMENT

We followed a set of methodological steps, leveraging various tools for efficient collaboration and project management. The key platforms utilized include:

### JIRA

We employed Jira for comprehensive project management. This allowed us to create and track tasks to ensure that the project progressed in a structured manner.

### GITHUB PRS (PULL REQUESTS):

The development and code review process were facilitated through Github. We utilized pull requests to propose and discuss changes, and made use of github actions for CI/CD tasks.

### OVERLEAF

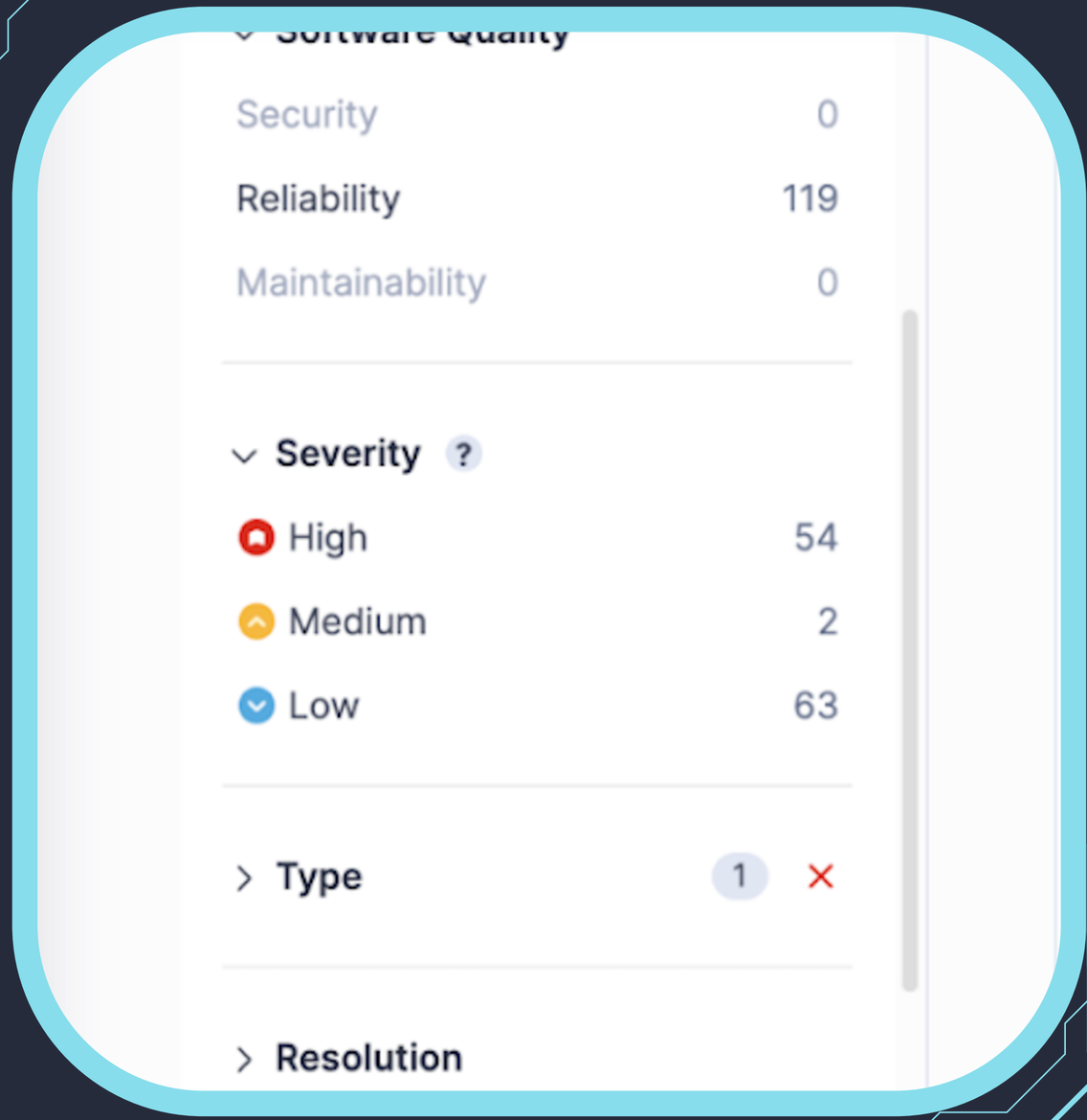
For the report writing process, we utilized Overleaf. This platform enabled us to collaboratively work on the report and upload changes through pull requests (PRs), ensuring that the team could contribute collectively in a structured manner

# APACHE PROJECT COMMONS-IMAGING

Apache Commons Imaging is an open-source Java library for image processing, supporting formats like JPEG, PNG, GIF, BMP, and TIFF. It prioritizes cross-platform compatibility and fosters collaboration, making it essential for Java developers seeking robust image processing capabilities.

# ANALYSIS SONARCLOUD

After analyzing the commons-imaging-dependability project on SonarCloud, we found the following issues in the code



# TYPES OF ERRORS

## HIGH - 54

- Errors are related to divisions.
- There are also errors of the type where referencing a static member of a subclass from its parent during class initialization makes the code more fragile and prone to future errors

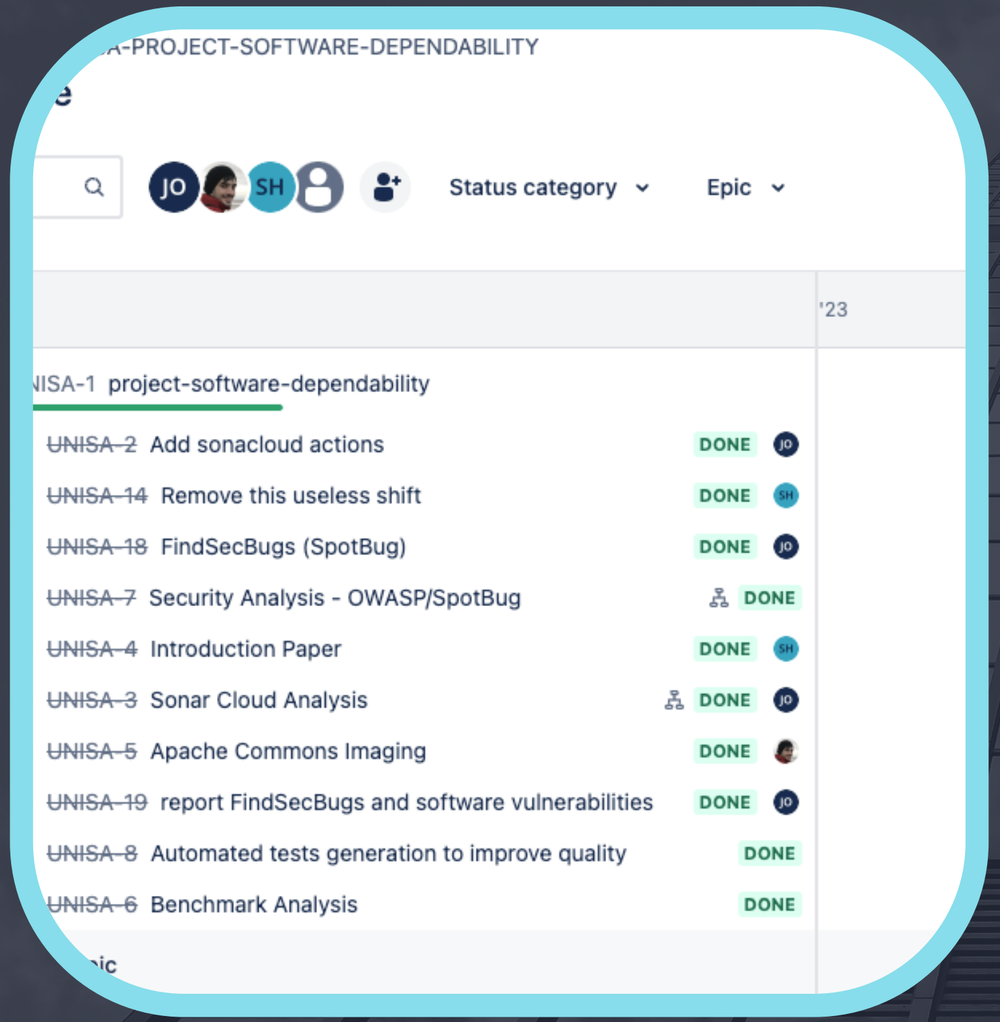
## MEDIUM - 2

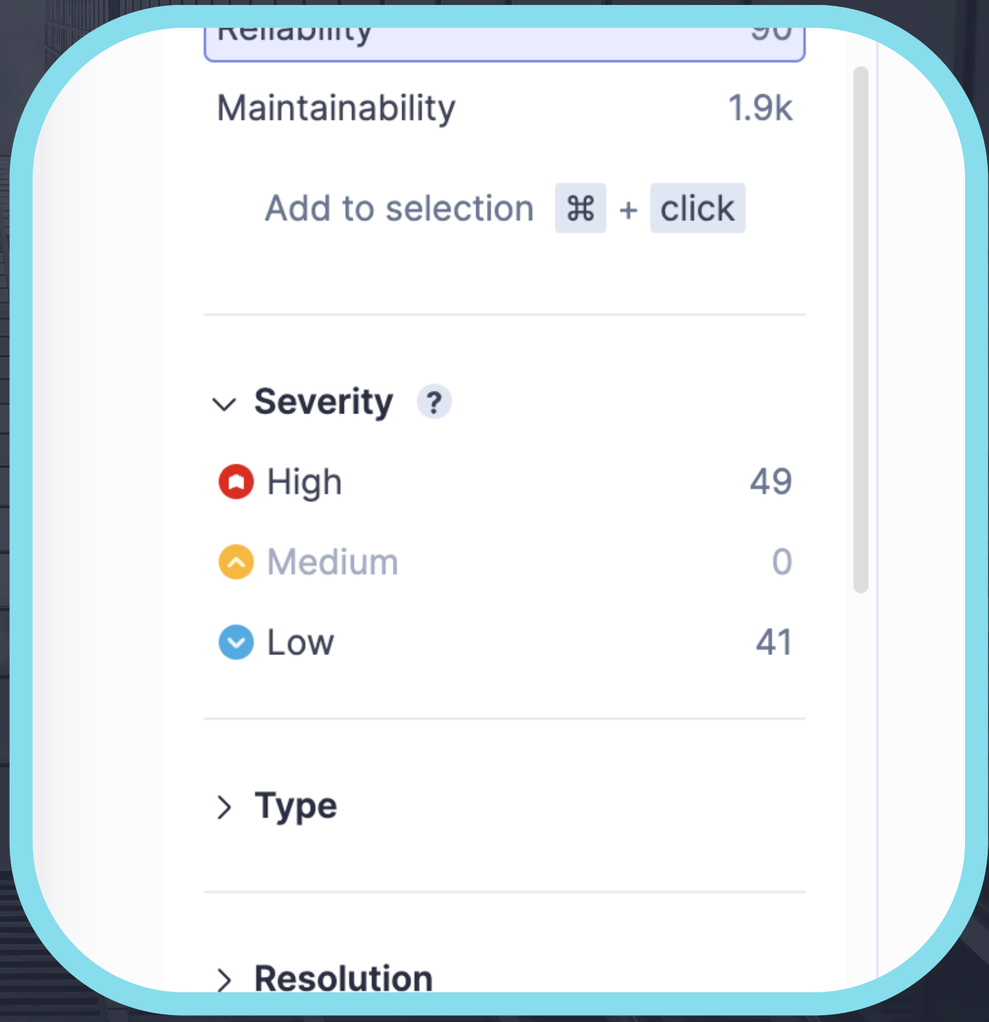
- Try to obtain the length of a pallet that could be null

## LOW - 63

- Suggestion to remove a left shift
- Additionally, there are some errors related to number casting

# BUGS FIXED | JIRA



# SOFTWARE TESTING

# JACOCO

## TOOL

- **JaCoCo** is an open-source toolkit for measuring code coverage in a code base and reporting it via visual reports.
- A notable feature of JaCoCo lies in its ability to set a minimum expected coverage

## RESULTS

- Instructions Coverage: 77%
- Branches Coverage: 63%
- Missed Cyclomatic Complexities: 2448/6213
- Missed Lines: 3931/17299
- Missed Methods: 511/2539
- Missed Classes: 16/431

# SOFTWARE TESTING

# CODECOV

## TOOL

- **CodeCov** is a web platform that provides insights regarding test coverage percentages for instructions and branches. It identifies missed complexities, lines, methods, and classes, helping maintain project robustness by setting minimum expected coverage levels for code changes.

## RESULTS

Main Project Results:

- Code Coverage: 71.17%
- Lines Covered:  12311/17299

After adding test cases for files with 0% of Coverage:

- Code Coverage: 71.82%
- Lines Covered: 12427/17303

# SOFTWARE TESTING
# PITEST

## TOOL

Pitest is an open-source mutation testing tool designed for Java projects and  it is aimed to identify areas of weakness in test coverage, guiding developers in enhancing the robustness of their test suites and, consequently, improving overall code quality and reliability.

## RESULTS

- Number of Classes: 286
- Line Coverage: 12771/16595 [77%]
- Mutation Coverage: 10217/15393 [66%]
- Test Strength: 10217/12277 [83%]

# SOFTWARE TESTING
# EVOSUITE

## TOOL

Evosuite is an open-source test generation tool for Java applications, specifically designed for generating unit tests automatically. It utilizes evolutionary algorithms to evolve and generate test cases that aim to achieve high code coverage and focuses on creating unit tests that exercise different paths and scenarios within the target code, helping developers discover potential bugs and vulnerabilities.

## RESULTS

**Default Coverage Criteria**

Code Coverage Boost: 73.9% to 76.3%

Tests Generated: 584

**Branch Coverage Criteria**

Code Coverage Boost: 73.9% to 77.6%

Tests Generated: 726

**Wilcoxon Results**

Wilcoxon Size: 0.015973411027052232

Wilcoxon Length: 0.008919317758844413

Wilcoxon Score: 0.20010965436042927

# SOFTWARE TESTING

## JMH

### TOOL

Java Microbenchmark Harness it’s a powerful tool designed for benchmarking Java code, allowing us to analyze the performance of our compression algorithms.

### DEMANDING IMAGE FORMATS

In the context Commons Imaging Project, an in-depth analysis of CPU-intensive image formats has revealed that TIFF and WebP were the most demanding ones.

### BENCHMARK TEST

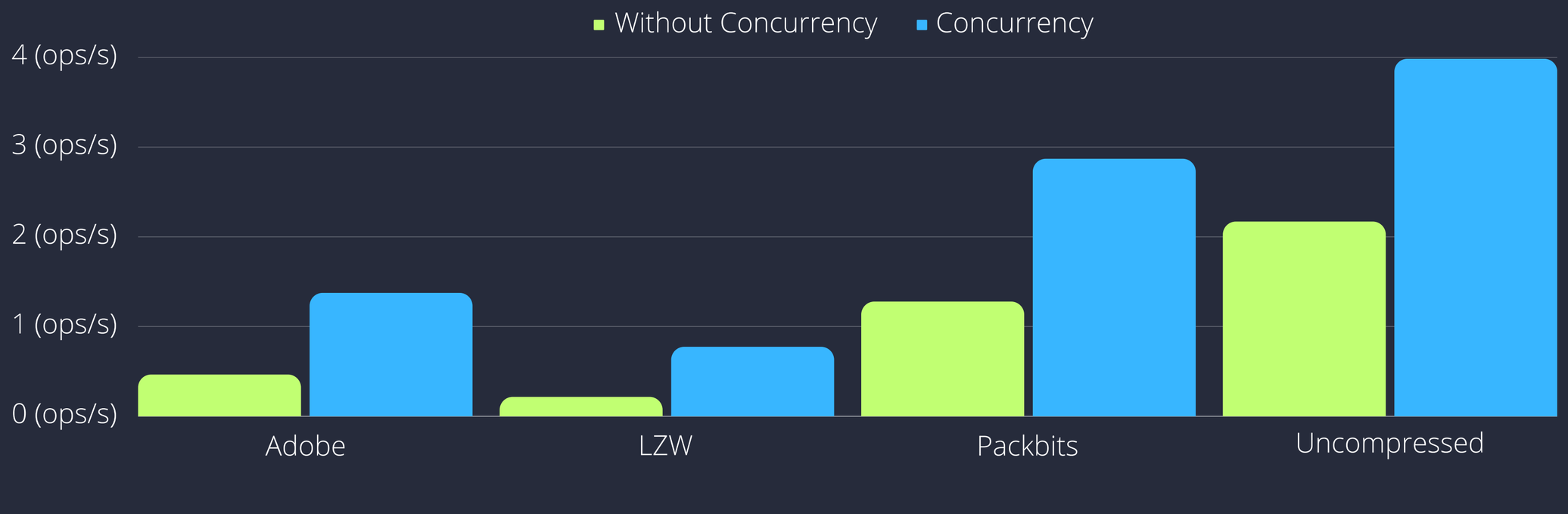
Test the compression of TIFF images using different algorithms, including LZW Compression, Packbits Compression, Adobe Compression, and Uncompressed.

### RESULTS

- ADEFLATE (Adobe): Throughput 0.465 ± 0.006 ops/s
- LZW: Throughput 0.216 ± 0.001 ops/s
- PACKBITS: Throughput 1.278 ± 0.053 ops/s
- Uncompressed: Throughput 2.170 ± 0.019 ops/s

# JMH RESULTS

Comparison not using Concurrency / Using Concurrency in test.



# SOFTWARE TESTING

## JMH

### IMPROVEMENT

Upon thorough review, we identified an opportunity for performance enhancement by introducing parallelism into the compression process.

### RESULTS

- DEFLATE (Adobe): Throughput 1.375 ± 0.012 ops/s (Increase by 196.77%)
- LZW: Throughput 0.774 ± 0.008 ops/s (Increase by 258.33%)
- PACKBITS: Throughput 2.870 ± 0.164 ops/s (Increase by 124.61%)
- Uncompressed: Throughput 3.984 ± 0.066 ops/s (Increase by 83.12%)

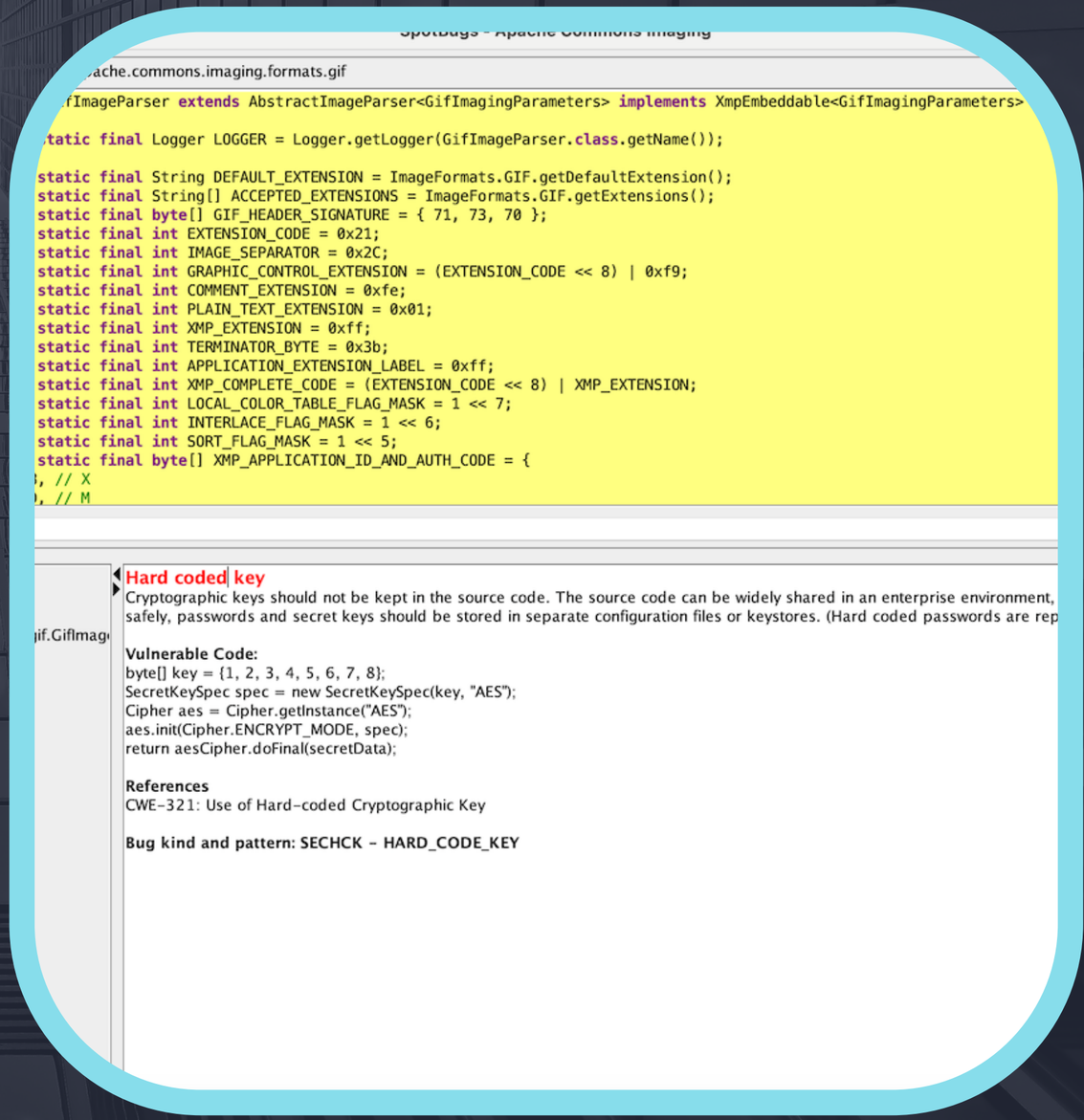
# SOFTWARE VULNERABILITIES

- **Data Injection:** This can occur when external input is not properly filtered or validated, allowing an attacker to inject malicious code.
- **Buffer Overflow:** This type of vulnerability occurs when the allotted storage capacity for a buffer is exceeded, enabling an attacker to overwrite surrounding memory.
- **Inadequate Validation:** Lack of input validation can lead to security issues. For example, if a user's authenticity is not verified before executing a critical operation.
- **Authentication and Authorization Issues:** Failures in au- thentication or authorization mechanisms may allow unauthorized users to access sensitive resources or functions.
- **Incorrect Error Handling:** Improper error handling can expose sensitive information or allow an attacker to discover weaknesses in the system

# ANALYSIS
# SPOTBUG

**FindSecBugs references these types of vulnerabilities as the inappropriate storage of cryptographic keys in the source code**

For secure management, passwords and secret keys should be stored in separate configuration files or keystores.



# SOFTWARE VULNERABILITIES
# OWASP DEPENDABILITY CHECKER

## TOOL

- **OWASP Dependency Checker:** The OWASP Dependency Checker serves as a crucial tool for software dependability analysis. Its primary purpose is to conduct a comprehensive examination of project dependencies, identifying potential vulnerabilities and ensuring the integrity of the software supply chain

## RESULTS

- **Dependencies Identified:**
  - commons-io-2.15.0.jar
  - commons-math3-3.2.jar
  - jmh-core-1.36.jar
  - jmh-generator-annprocess-1.36.jar
  - jopt-simple-5.0.4.jar

- **Analysis Results:** All dependencies show high-confidence levels according to OWASP Dependency Checker.

# CONCLUSION

The Apache Commons Imaging project benefits from a comprehensive testing strategy, utilizing CodeCov, Pitest, and JMH benchmarks to enhance code coverage and performance. Evosuite's automated prowess, particularly with Branch Coverage Criteria, reinforces the project's testing robustness. The project's dependency management, validated by OWASP, ensures robustness, and FindSecBugs emphasizes security, guiding the project toward a resilient, well-rounded software solution.