Apache Commons-email

Luigi Allocca, Simone Della Porta, Rocco Iuliano

ACM Reference Format:
Luigi Allocca, Simone Della Porta, Rocco Iuliano. 2023. Apache Commons-email. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The aim of this project is to analyze and improve an existing project by Apache foundation. We will improve the availability (i), reliability (ii), safety (iii), security (iv) and resilience (v) of the project in order to improve the dependability. We will focus on improve security by improving the coverage of the test cases and creating new ones, fix the bugs related to security and reduce security hotspot issues which refer to a specific area or component of a software system that has a higher risk of security vulnerabilities or breaches. These issues are often identified through a security analysis or review, and they can pose a significant threat to the overall security of the system. By removing these security hotspots, the overall security of the project will be improved, reducing the risk of security breaches and protecting sensitive user data. This can lead to increased trust from users and stakeholders, and can help to ensure that the project is compliant with relevant security regulations and standards. Overall the general goals are:

- fix as many project bugs as possible, prioritizing the crucial bugs;
- improve the coverage of project testing by developing new test cases and improving the existing ones;
- minimize the number of code smells;
- reduce security hotspot issues;
- verify the project performance.

2 PROJECT PRE REQUIREMENTS

We choose a project with the following characteristics in order to improve the dependability of the software with CI/CD paradigm using the tools introduced in the course:

- Git Actions to check code quality, coverage, Java CI and security.
- The project should use Maven to manage the project build, so it should have the pom.xml file.

3 CONTEXT OF THE PROJECT

Apache Commons-Email is an open source project and it is released in 01/08/2017 and it is available on GitHub to the following link: https://github.com/apache/commons-email. It aims to provide a API for sending email. Apache Commons-Email is built on top of the Java Mail API, which it aims to simplify. This project is composed by twentyfour contributors and has got 78 fork, 5 branches and 1132 commits of which the last commit was made in 18/03/2023 (id commit: f41dd76). Moreover, Apache Commons-Email has got three type of actions:

(1) **CodeQL** that allows us to verify the quality of the project;
(2) **Coverage** that allows us to verify the per cent of coverage of project test cases;
(3) **Java CI** that allows us to apply all the test cases to each commit that we make, so we are sure that the changes that we applied to the software doesn't introduce a bug.

Apache Commons-Email has got a user guide and a JavaDoc API documentation that are available to the following link:

- User guide: https://commons.apache.org/proper/commons-email/userguide.html
- JavDoc: https://commons.apache.org/proper/commons-email/javadocs/api-release/index.html

Some popular Java-based applications that use Apache Commons Email are Apache Jenkins, Apache OFBiz, and Apache Syncope. Additionally, many Java-based web applications and enterprise systems use Apache Commons Email to handle email sending functionality.
To apply the changes to the project we must follow this rules:

(1) No tabs, instead use spaces for indentation.
(2) Respect the code style.
(3) Create minimal diffs - disable on save actions like reformat source code or organize imports. If you feel the source code should be reformatted create a separate PR for this change.
(4) Provide JUnit tests for your changes and make sure your changes don't break any existing tests by running mvn.

4 PRELIMINAR ANALISYS

After selecting the project Commons-email, we have created a fork of the repository, cloned the repository and built the project in order to run all test cases which result passed successfully. Then we performed a test push to verify the project actions and check the reults.

5 METRICS

6 METHODOLOGICAL STEPS CONDUCTED TO ADDRESS THE GOALS

Then we have conducted a preliminar analisys of the project by using *sonarcloud* [1] and *codecov* [2] and we obtained this results:

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

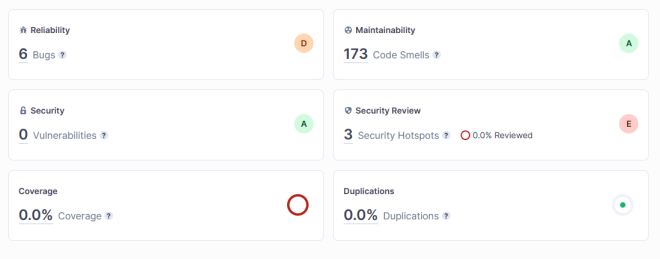

Figure 1: Sonarcloud analysis

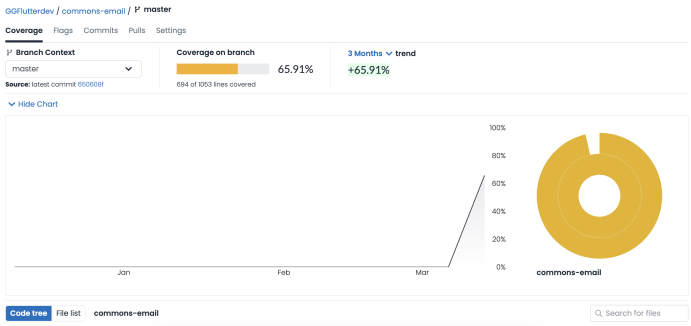

Figure 2: Codecov analysis

The result of the analysis are:

(1) The coverage is 65,91%;
(2) 6 bugs, of which:
- 2 Major problems relative to synchronization.
- 2 Major problems relative to NullPointerException.
- 2 Critical problems relative to bad formatted try/catch.

(3) 173 code smells, of witch:
- 11 Bloker problems related to the adding of at least one assertion in test cases
- 3 Critical problems related to the try-catch implementation
- 5 Critical problems related to the Cognitive Complexity allowed
- 7 Critical problems related to duplicate literals
- 7 Major problem related to commented out code
- 42 Major problems related to Assert Equals and Assert Not Equals
- 4 Major problems related to expressions that are always evaluated to `true`
- 4 Major relative to lambda exceptions
- 7 Major problems related to the use of a generic exeption instead of create and use a dedicated exception
- 3 Major problems relative to testing
- 3 Major problems related to swap arguments
- 2 Major problems related to the visibility of one constructor
- 2 Major problem related to the try-catch implementation
- 1 Major problem related to the number of assertions in a method
- 1 Major problem related to nested block of code
- 1 Major problem related to the usage of a method instead of the usage of another method
- 1 Major problem related to the usage of the utility of System class instead of the introdiction of a logger
- 14 Minor problems related to deprecated code
- 2 Mimonr problems related to the chatset name argument
- 2 Minor problems related to hard-coded path-delimiter
- 2 Minor problem related to the try-catch implementation
- 1 Minor problem related to a return statement
- 48 Info problems related to deprecated code

(4) 3 security hotspots of which:
- 2 related to DoS
- 1 related to weak cryptography

So for having a continuous analisys of the project we decided to integrate these tools in the our project in order to analyze the code after every push or pull request.

7 RESULTS AND FINDINGS

8 CONCLUSIONS