Secure Software Engineering Static Analysis Tools

Introduction

SonarQube is a self-managed, automatic code review tool that systematically helps you deliver Clean Code. As a core element of our Sonar solution, SonarQube integrates into your existing workflow and detects issues in your code to help you perform continuous code inspections of your projects. The product analyses 30+ different programming languages and integrates into your Continuous Integration (CI) pipeline of DevOps platforms to ensure that your code meets high-quality standards. Sonarqube also ensures code reliability, Application security, and reduces technical debt by making your code base clean and maintainable. Sonarqube also provides support for 27 different languages, including C, C++, Java, Javascript, PHP, GO, Python, and much more. SonarQube also provides Ci/CD integration and gives feedback during code review with branch analysis and pull request decoration.

Clean Code is the standard for all code that results in secure, reliable, and maintainable software therefore, writing clean code is essential to maintaining a healthy codebase. This applies to all code: source code, test code, infrastructure as code, glue code, scripts, and more. For details, see Clean Code. Sonar's Clean as You Code approach eliminates many of the pitfalls that arise from reviewing code at a late stage in the development process. The Clean as You Code approach uses your quality gate to alert/inform you when there's something to fix or review in your new code (code that has been added or changed), allowing you to maintain high standards and focus on code quality.

Developing with Sonar

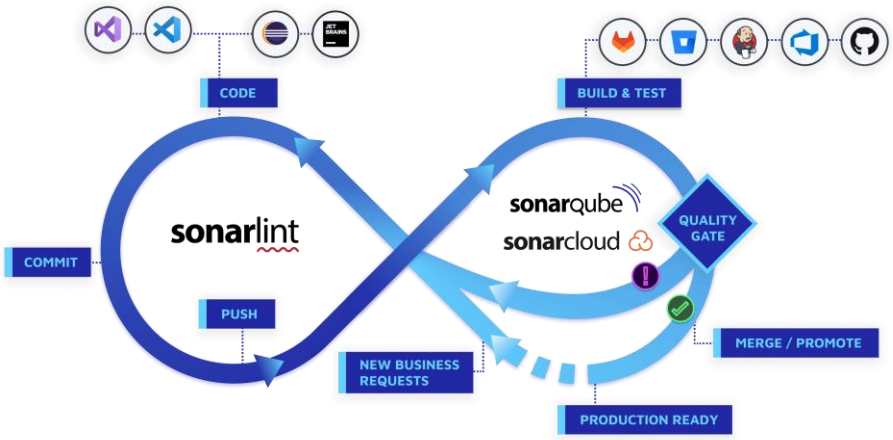

Figure 1: SonarQube Code Review Lifecycle and CI integration.

The Sonar solution performs checks at every stage of the development process:

- SonarLint provides immediate feedback in your IDE as you write code so you can find and fix issues before a commit.
- SonarQube's PR analysis fits into your CI/CD workflows with SonarQube's PR analysis and use of quality gates.
- Quality gates keep code with issues from being released to production, a key tool in helping you incorporate the Clean as You Code methodology.
- The Clean as You Code approach helps you focus on submitting new, clean code for production, knowing that your existing code will be improved over time.

Organizations start off with a default set of rules and metrics called the Sonar way quality profile. This can be customized per project to satisfy different technical requirements. Issues raised in the analysis are compared against the conditions defined in the quality profile to establish your quality gate.

A quality gate is an indicator of code quality that can be configured to give a go/no-go signal on the current release-worthiness of the code. It indicates whether your code is clean and can move forward.

- A passing (green) quality gate means the code meets your standard and is ready to be merged.
- A failing (red) quality gate means there are issues to address.

SonarQube provides feedback through its UI, email, and in decorations on pull or merge requests (in commercial editions) to notify your team that there are issues to address. Feedback can also be obtained in SonarLint supported IDEs when running in connected mode. SonarQube also provides in-depth guidance on the issues telling you why each issue is a problem and how to fix it, adding a valuable layer of education for developers of all experience levels. Developers can then address issues effectively, so code is only promoted when the code is clean and passes the quality gate.

SonarQube is considered a Code Quality Assurance tool that collects and analyzes source code and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. Everything from minor styling choices to design errors are inspected and evaluated by SonarQube. This provides users with a rich searchable history of the code to analyze where the code is messing up and determine whether it is styling issues, code defeats, code duplication, lack of test coverage, or excessively complex code. The software will analyze source code from different aspects and drills down the code layer by layer, moving module level down to the class level, with each level producing metric values and statistics that should reveal problematic areas in the source code that needs improvement.

What is Code Quality?

Code quality is an indicator about how quickly developers can add business value to software systems.

Software Quality Characteristics: ISO/IEC 9126

To evaluate software, it is necessary to select relevant quality characteristics. ISO/IEC 9126 defines a quality model which is applicable to every kind of software. It defines six product quality characteristics.

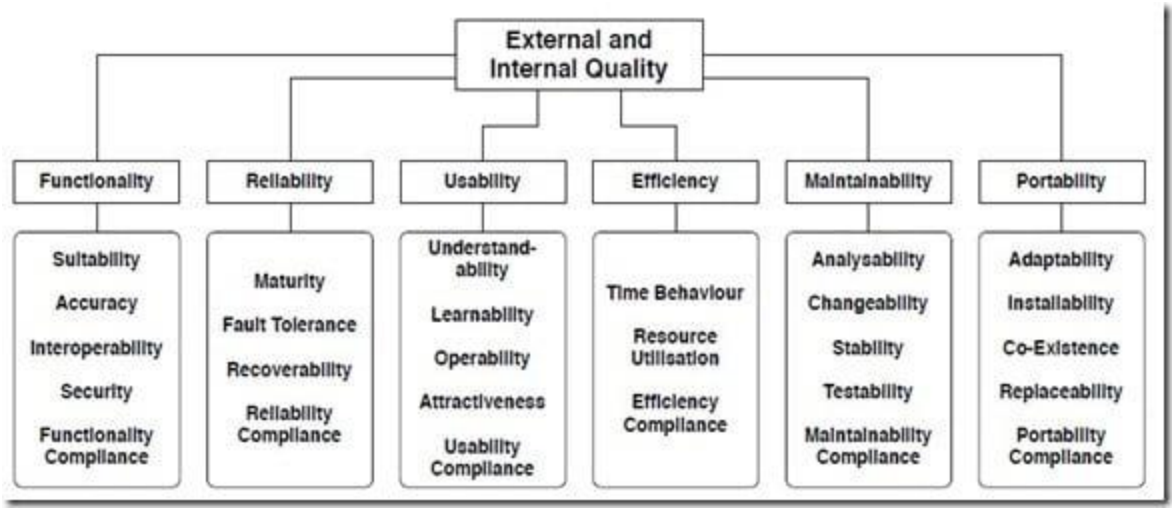

Figure 2: six product quality characteristics defined by ISO/IEC 9126 quality model.

What is SonarQube?

Sonar is an open-source software quality platform. SonarQube saves the calculated measures in a database and showcases them in a rich web-based dashboard. Provides trends and leading indicators.

How Sonar Works?

Sonar uses various static & dynamic code analysis tools such
as Checkstyle, PMD, FindBugs, FxCop, Gendarme, and many more to extract software metrics, which then can be used to improve software quality. Provides lots of plugins.

SonarQube CI

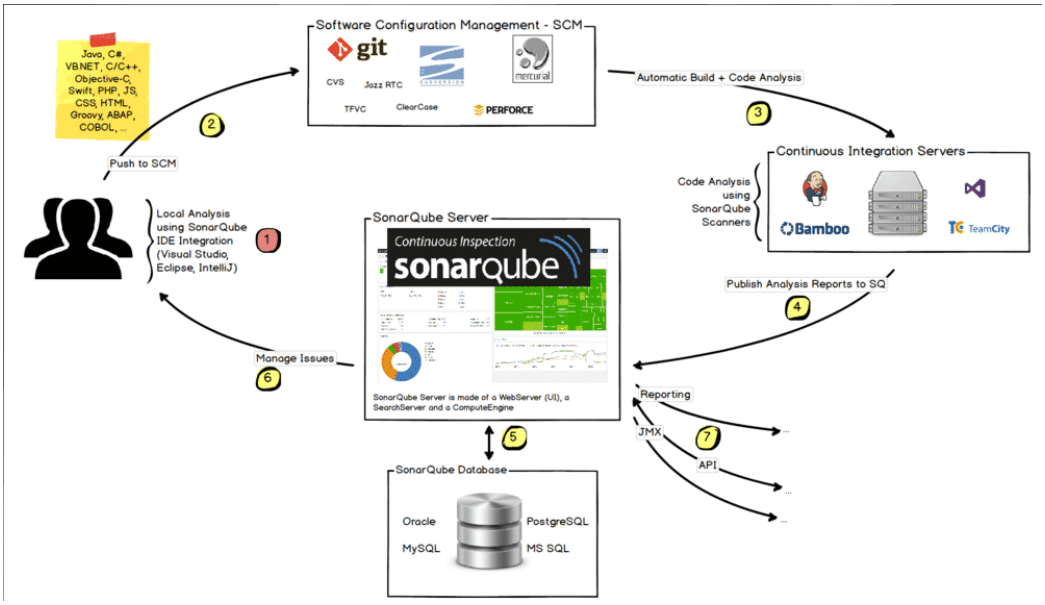

Figure 3: SonarQube Ecosystem and CI integration.

SonarQube Features

- Supports languages: Java, C/C++, Objective-C, C#, PHP, Flex, Groovy, JavaScript, Python, PL/SQL, COBOL, etc.
- Can also be used in Android development.
- Doesn't just show you what's wrong, but also offers quality and management tools to actively help you correct issues.
- Focuses on more than just bugs and complexity and offers more features to help the programmers write code, such as coding rules, test coverage, de-duplications, API documentation, and code complexity all within a dashboard.
- Gives a moment-in-time snapshot of your code quality today, as well as trends of past and potentially future quality indicators. Also provides metrics to help you make the right decisions.
- Offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, potential bugs, comments, design, and architecture.
- Records metrics history and provides evolution graphs ("time machine") and differential views.
- Provides fully automated analyses integrates with Maven, Ant, Gradle, and continuous integration tools (Atlassian Bamboo, Jenkins, Hudson, etc.).
- Integrates with the Eclipse. VS code, IntelliJ IDEA development environments.

- Integrates with external tools: JIRA, Mantis, LDAP, Fortify, etc.
- Is expandable with the use of plugins.

There are a few tiers to SonarQube, which will depend on how much you want the software to do and for what level of development you want to do with the software. A brief breakdown is as follows:

- Community edition: The starting point for adopting code quality in CI/CD.
- Developer Edition: Maximum Application security, and value from SonarQube across branches and PRS
- Enterprise Edition: Manage application portfolio, enable code quality and security at an enterprise level.
- Data Center Edition: High Availability for global deployments.

SonarQube Server installation

Installation roadmap

Proceed as follows to install SonarQube on server side:

1. Install the SonarQube database.
2. Install the SonarQube server and perform a basic setup. You can install the server either from the ZIP file or from the Docker image.
3. If necessary, perform advanced setup.

Instance components:

A SonarQube instance comprises three components:

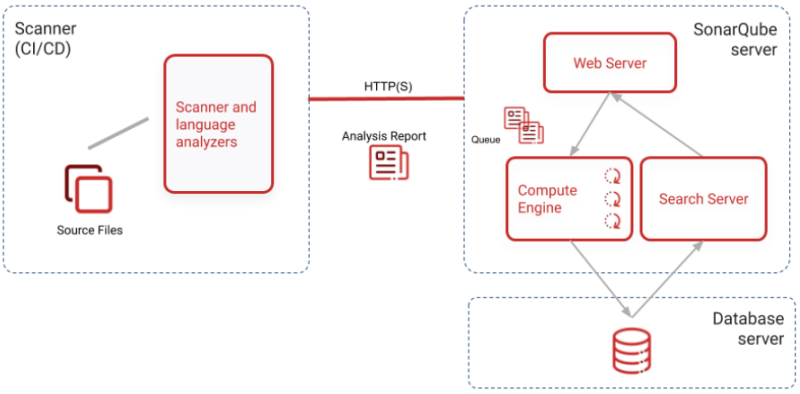

Figure 4: SonarQube instance components

Architecture and Integration

The SonarQube platform comprises of 4 components:

1. SonarQube Server comprising of the following processes:
    1. Web Servers for developers which allows managers to browse quality snapshots and configure the SonaQube instances.
    2. Search Server based on Elastic search to back searches from the UI.
    3. Compute engine server in charge of processing code analysis reports and saving them in the SOnarQube database.
2. SonarQube Database which stores:
    1. The configuration of the SonarQube instance
    2. The quality snapshots of projects, views, etc.
3. SonarQube plugins installed on the server, possibly including language, SCM, integration, authentication, and governance plugins.
4. SonarScanners running on your Build/Continuous integration servers to analyze a project

About machines and locations:

- The SonarQube platform cannot have more than one SonarQube server (unless clustered), and one SonarQube database.
- For optimal performance, each component should be installed on separate machines and the server machine should be dedicated.
- SonarScanners scale by adding machines.
- All machines must be time-synchronized.
- SonarQube server and SonarQube Database must be located in the same network
- SonarScanners don’t need to be on the same network as the SonarQube Server
- There is no communication between SonarScanners and the SonarQube Database.

To integrate SonarQube on a production environment, follow these steps:

1. Developers code in their IDEs and use SonarLint to run local analysis.
2. Developers push their code to their favorite SCM.
3. The Continuous Integration Server triggers an automatic build, and the execution of the Sonar Scanner required to run the SonarQube Analysis.
4. The analysis report is sent to the SonarQube Server for processing.

5. SonarQube server processes and stores the analysis report results in the SonarQube Database and displays the results in the UI.
6. Developers review, comment, challenge their issues to manage and reduce their Technical Debt through SonarQube UI.
7. Managers receive reports from the analysis, Ops use APIs to automate the configuration and extract data from SonarQube, and ops use JMX to monitor SonarQube Servers.

Installing a Plugin

There are two options to install a plugin into SonarQube: marketplace and manual installation

To install a plugin from the Marketplace

If you have access to the internet and you are connected with a SonarQube user having the Global Permission "Administer System" you can go to Administration > Marketplace (To learn more about the marketplace, go here: https://docs.sonarqube.org/latest/instance-administration/marketplace/)

Then you can find the plugin you want to install from the marketplace, and click on install and wait for the download to be processed. Once the download is complete, a restart button will be available to restart your instance.

To manually install a plugin

In the page dedicated to the plugin, you want to install (IE SonarPython), click on the "Download" link of the version compatible with your version of SonarQube, then put the downloaded jar file in $SONARQUBE_HOME/extensions/plugins, removing any previous versions of the same plugin that may be present. Once done you will need to restart SonarQube.

1. The SonarQube server running the following processes:
    - A web server that serves the SonarQube user interface.
    - A search server based on Elasticsearch.
    - The compute engine in charge of processing code analysis reports and saving them in the SonarQube database.
2. The database to store the following:
    - Metrics and issues for code quality and security generated during code scans.
    - The SonarQube instance configuration.
3. One or more scanners are running on your build or continuous integration servers to analyze projects.

Hosts and locations

For optimal performance, the SonarQube server and database should be installed on separate hosts, and the server host should be dedicated. The server and database hosts should be located on the same network. All hosts must be time-synchronized.

Installing the SonarQube database

Several external database engines are supported:

1. Microsoft SQL Server
2. Oracle
3. postgreSQL

Create an empty schema and a sonarqube user. Grant this sonarqube user permissions to create, update, and delete objects for this schema.

Install SonarQube Server in Local Machine

To install SonarQube locally to analyze a project, you have two different methods: Installing SonarQube from a zip file, or from a docker image.

Getting SonarQube from a Zip file

- You can download the SonarQube community edition zip file from here: https://www.sonarqube.org/downloads/
- As a non-root user, unzip it in any file location you want it in, such as c:/sonarqube or /opt/sonarqube
- Start the SonarQube server:
  on windows: C:\sonarqube\bin\windows-x86-64\StartSonar.bat
  On other operating systems: /opt/sonarqube/bin/[OS]/sonar.sh console
- Login to **http://localhost:9000** with the following credentials for system admin: (admin/admin)

Getting SonarQube from a Docker Image

- Find the community version of SonarQube that you want to use on Docker Hub: https://hub.docker.com/_/sonarqube/
- Start the server by running:

docker run -d --name sonarqube -p 9000:9000 <image name>

- Login to **http://localhost:9000** with the following credentials for system admin: (admin/admin)

Analyzing source code

Once the SonarQube platform has been installed, you're ready to install a scanner. You must install and configure the scanner that is most appropriate for your needs depending on your build tool. Then, you will start the analysis of your project by invoking the scanner in your CI pipeline after you build and before you do any sort of archiving or deployment. The scanner installation and invoking are explained in detail in the documentation of the corresponding scanner:

- Gradle - SonarScanner for Gradle
- .NET - SonarScanner for .NET
- Maven - SonarScanner for Maven
- Jenkins - SonarQube extension for Jenkins
- Azure DevOps - SonarQube Extension for Azure DevOps
- Ant - SonarScanner for Ant
- Anything else - SonarScanner CLI

You may use one of the following options:

- You install the scanner on your CI/CD host and, when you start your first analysis, the corresponding project is automatically created in SonarQube.
- You create first a project in the SonarQube UI. During the creation, you will be asked about the CI tool you are using, and you will be guided through a tutorial to install the appropriate scanner on your CI/CD host.

To create a project, you use the **Create Project** button on the **Projects** page, which is visible to users with project creation rights.

SonarQube integrations are supported for popular DevOps Platforms: GitHub Enterprise and GitHub.com, BitBucket Server, Azure Devops Server and Azure DevOps Services.

What does analysis produce?

SonarQube can analyze up to 29 different languages depending on your edition. The outcome of this analysis will be quality measures and issues (instances where coding rules were broken). However, what gets analyzed will vary depending on the language:

- On all languages, "blame" data will automatically be imported from supported SCM providers. Git and SVN are supported automatically. Other providers require additional plugins.
- On all languages, a static analysis of source code is performed (Java files, COBOL programs, etc.)

- For certain languages, the static analysis should be done on compiled code (.class files in Java, .dll files in C#, etc.)

Will all files be analyzed?

By default, only files that are recognized by your edition of SonarQube are loaded into the project during analysis. For example, if you're using SonarQube Community Edition, which includes analysis of Java and JavaScript, but not C++, all .java and .js files would be loaded, but .cpp files would be ignored.

What about branches and pull requests?

Developer Edition adds the ability to analyze your project's branches and pull requests as well as the ability to automatically report your pull request analysis to your DevOps platform interface.

What happens during analysis?

During analysis, data is requested from the server, the files provided to the analysis are analyzed, and the resulting data is sent back to the server at the end in the form of a report, which is then analyzed asynchronously server-side.

Analysis reports are queued and processed sequentially, so it is quite possible that for a brief period after your analysis log shows completion, the updated values are not visible in your SonarQube project. However, you will be able to tell what's going on because an icon will be added on the project homepage to the right of the project name. Mouse over it for more detail (and links if you're logged in with the proper permissions).

Analyzing a project with SonarQube

Once you are logged into sonarqube, to analyze a project follow the following steps:

1. Click “create new project” button.
2. When asked How do you want to create your project, select Manually.
3. Give your project a Project key and a Display name and click the Set Up button.
4. Under Provide a token, select Generate a token. Give your token a name, click the Generate button, and click Continue.
5. Select your project’s main language under Run analysis on your project and follow the instructions to analyze your project. Here you’ll download and execute a Scanner on your code (if you’re using Maven or Gradle, the Scanner is automatically downloaded).

After successfully analyzing your code, you’ll see your first analysis on SonarQube.

Part 2: Demonstration of SonarQube Tool

Scope

Used SonarQube to conduct code review on an open-source application Juice-Shop by OWASP. Juice-Shop is an insecure application made by OWASP for software security testing and demonstration.

Results

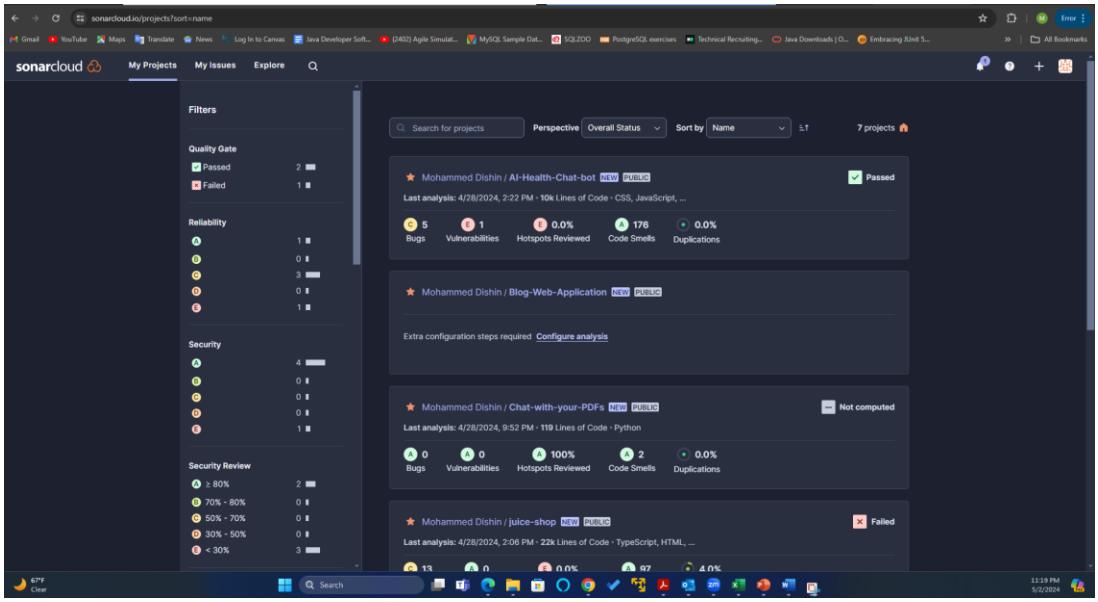

Figure 5: showing SonarQube main dashboard.

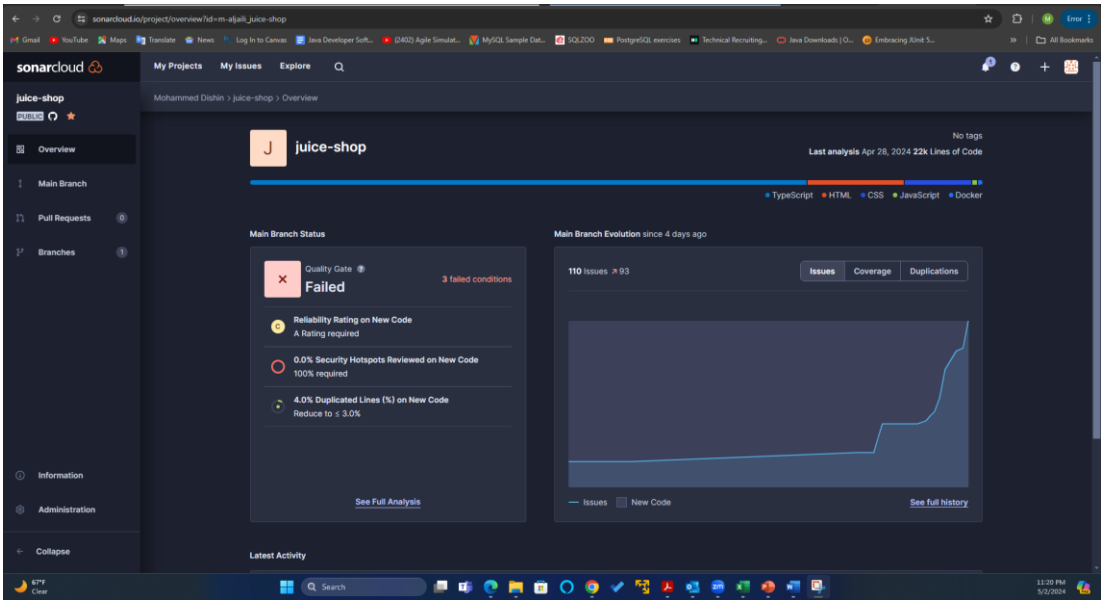

Figure 6: Showing project review summary report.

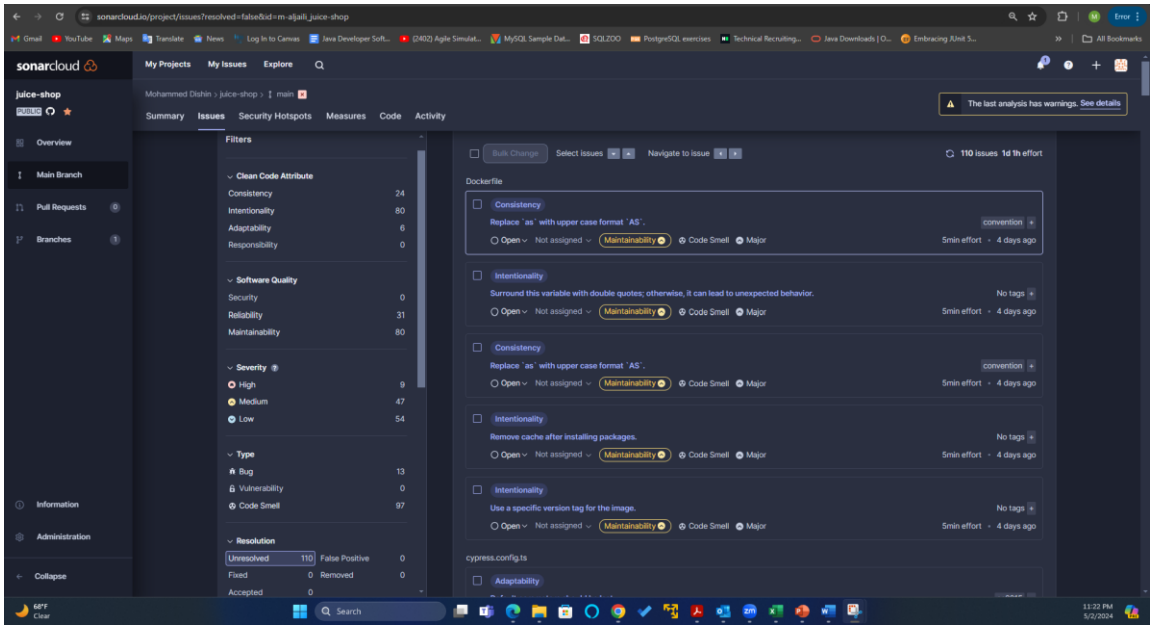

Figure 7: Detailed dashboard for review report by SonarQube.

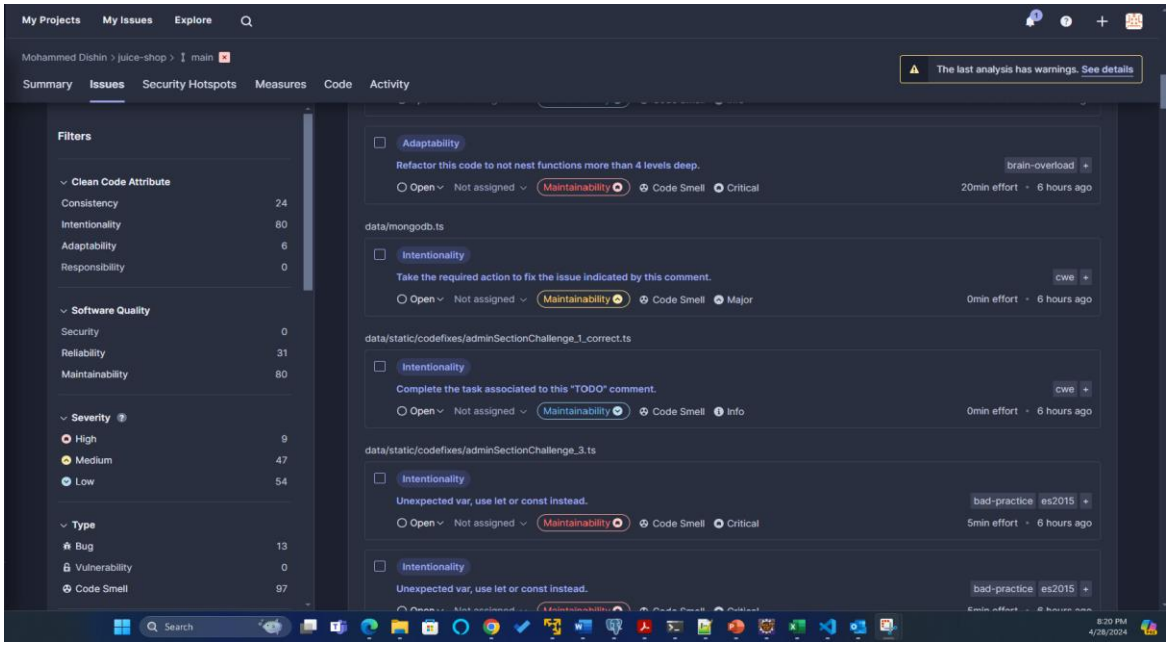

Figure 8: showing type and severity of reported issues.

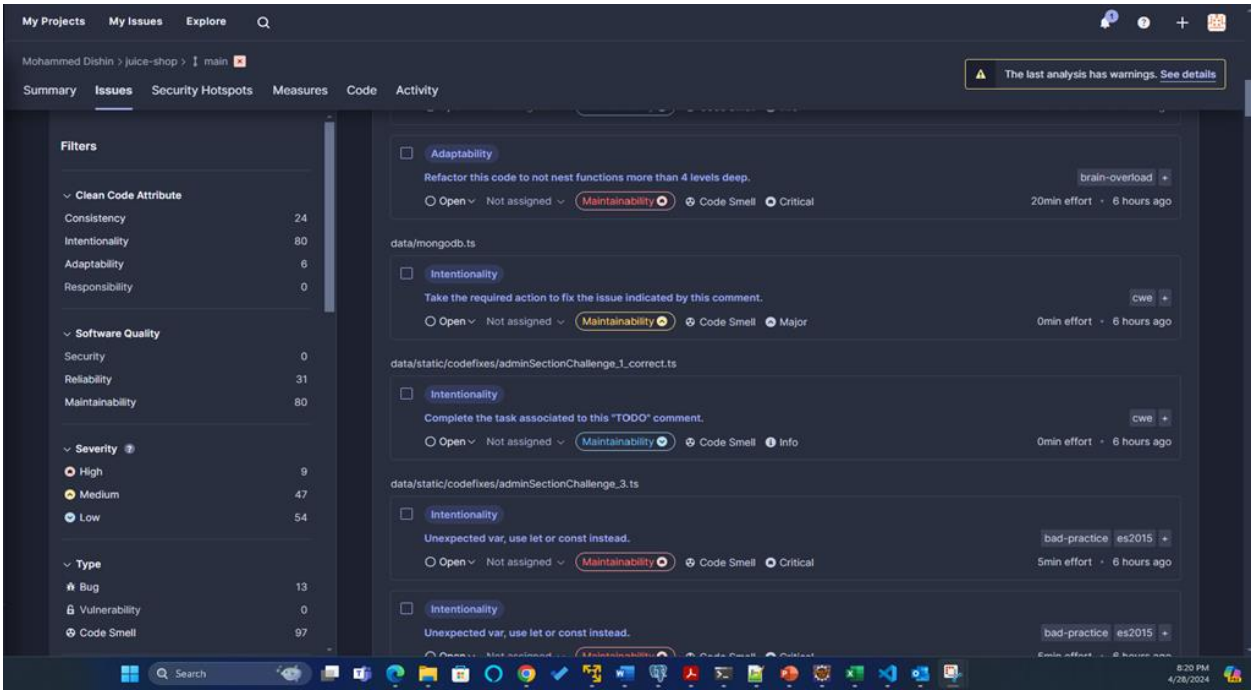

Figure 9: Showing status of reported issues and links to dig deep.

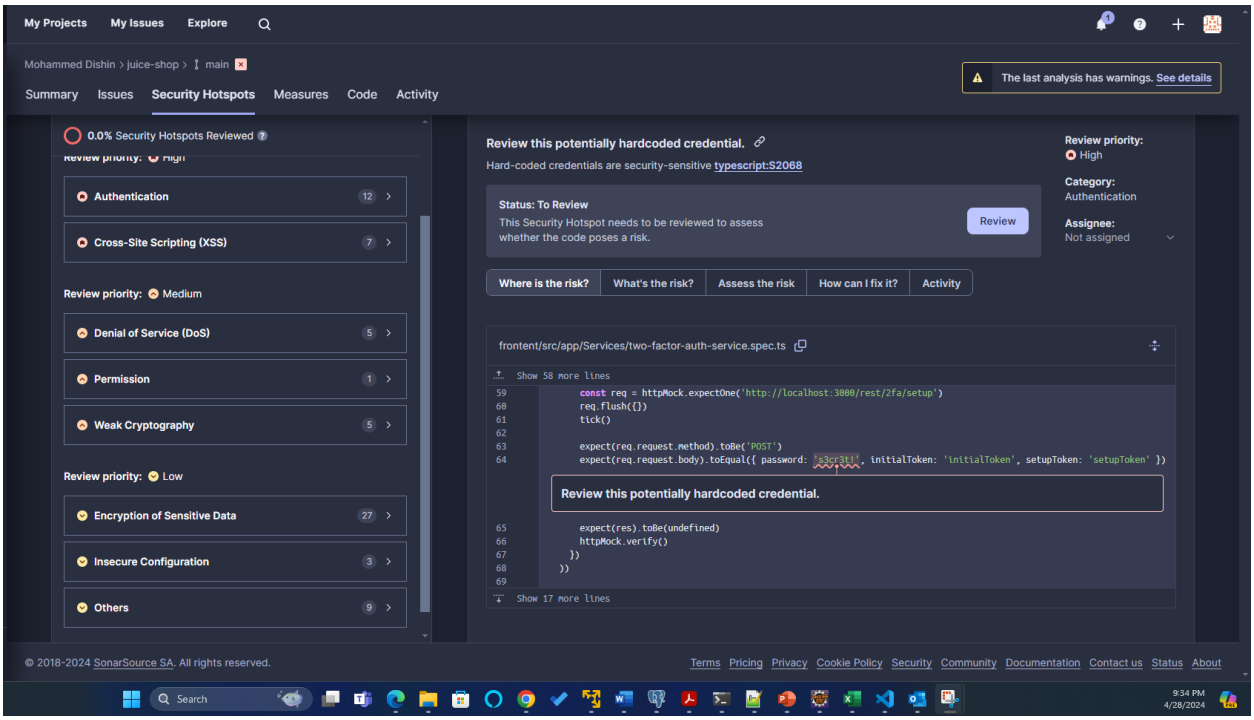

Figure 10: showing details of reported issue and how it could be resolved.

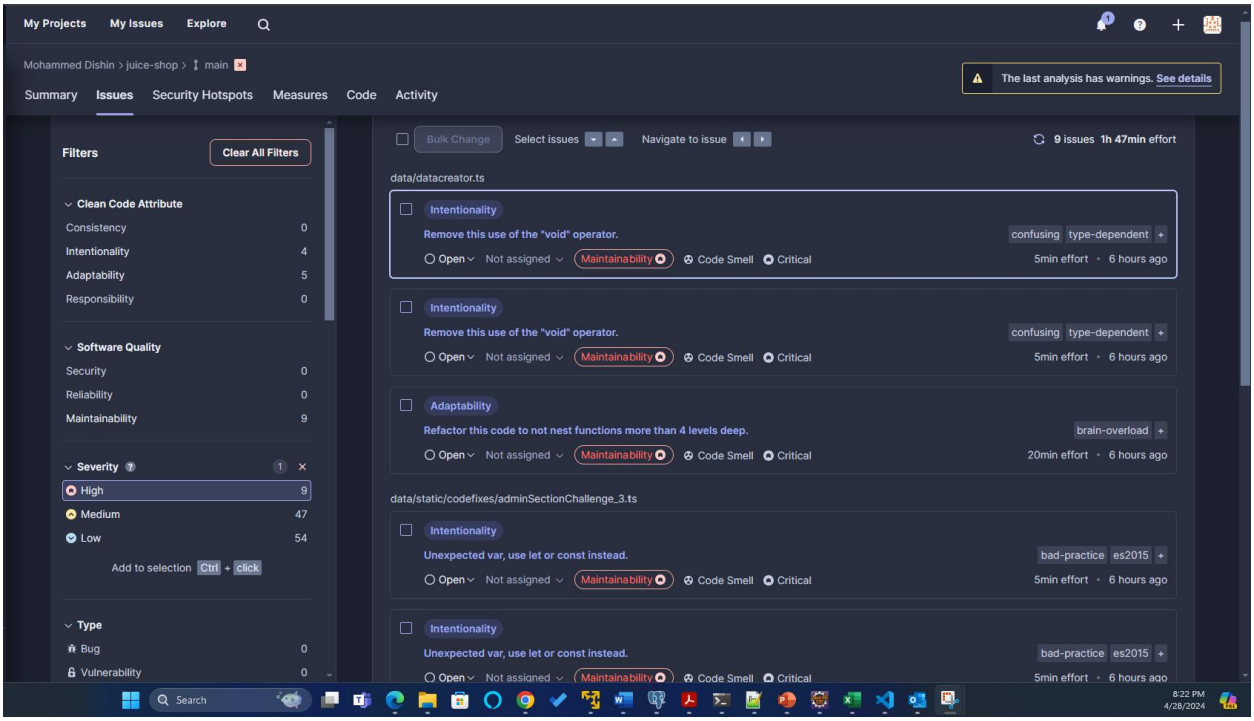

Figure 11: Filtering only High severity issues.

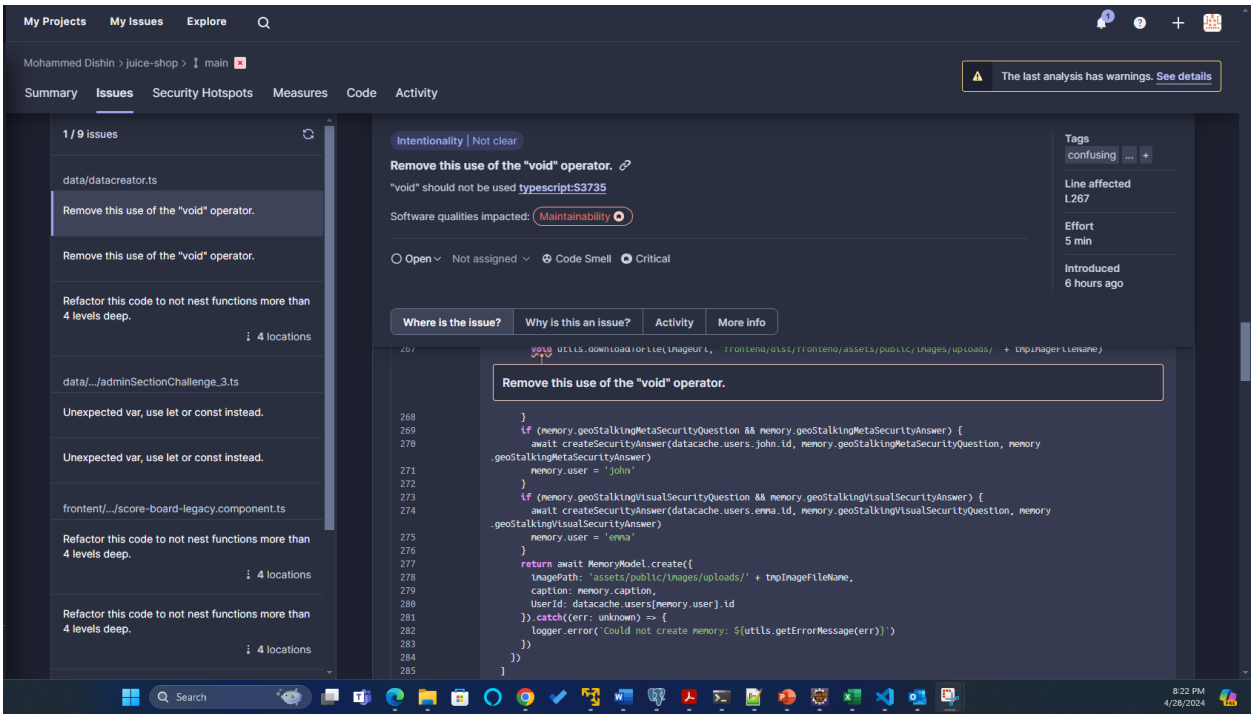

Figure 12: showing suggestion to resolve the reported issue.

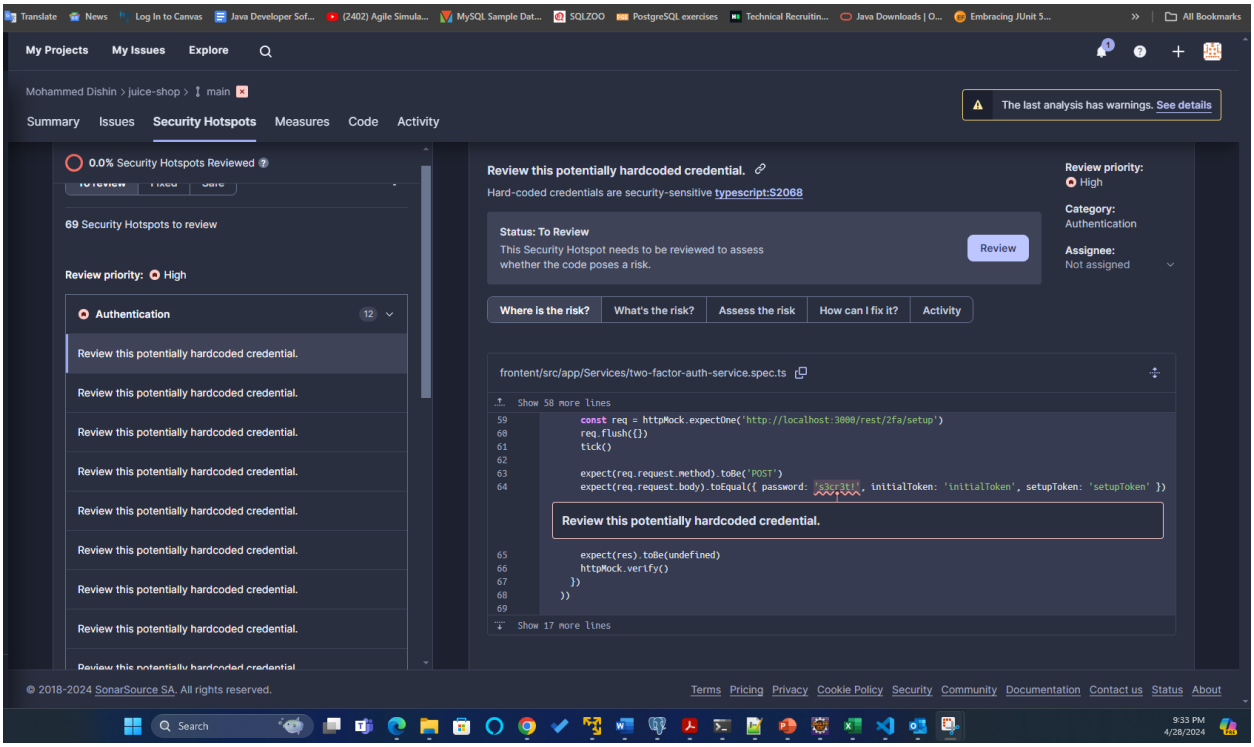

Figure 13: multiple authetication vulnerabilties reported.

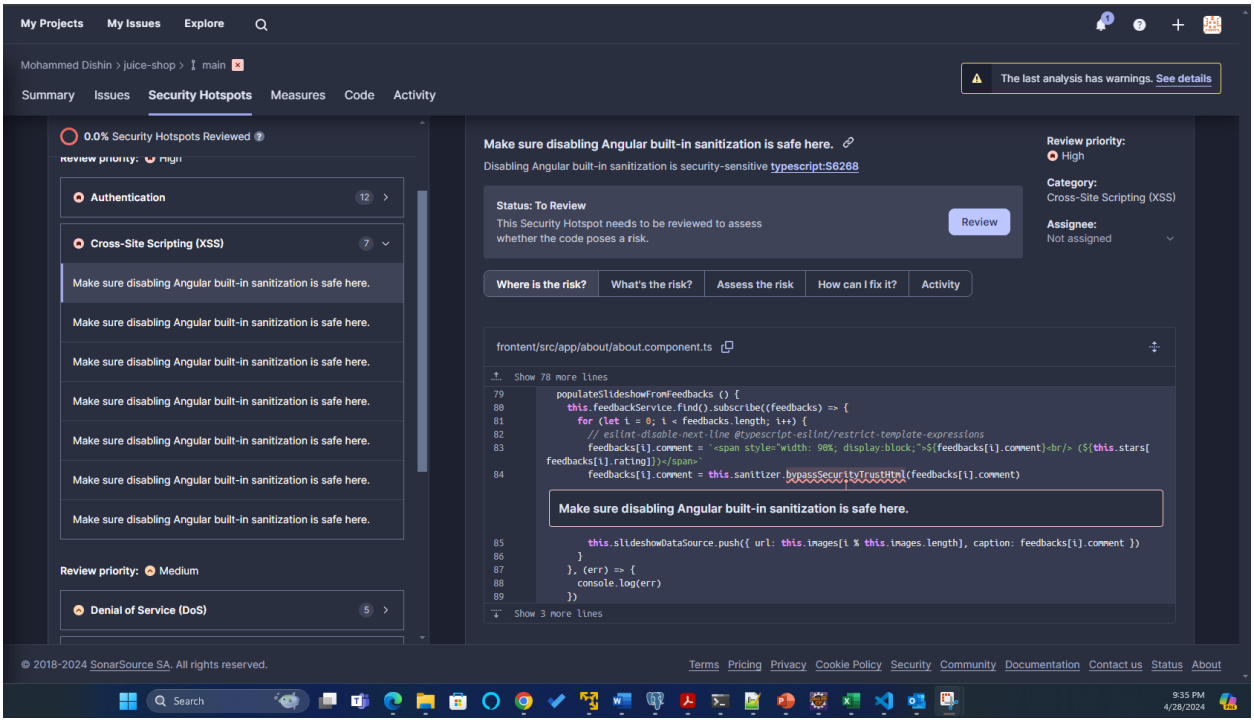

Figure 14: Corss-Site Scripting XSS vulnerability detected.

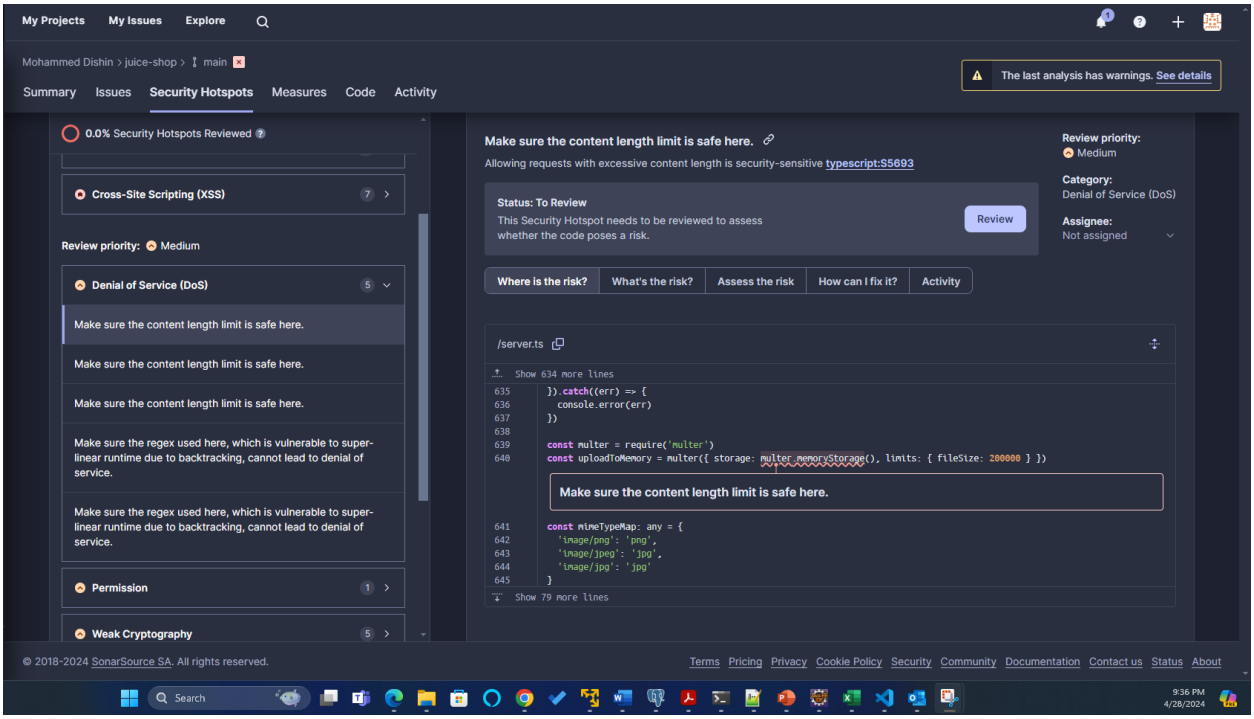

Figure 15: Buffer Overloading vulnerability detected.

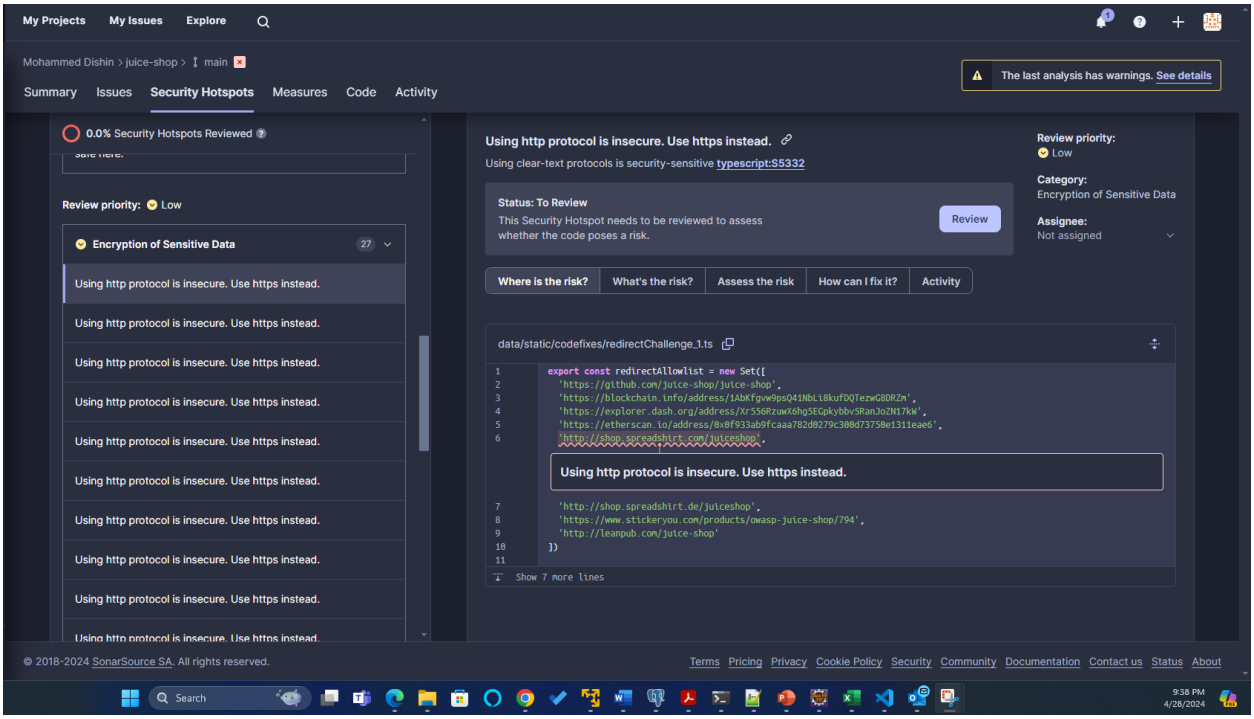

Figure 16: Insecure HTTP Protocol vulnerability detected.

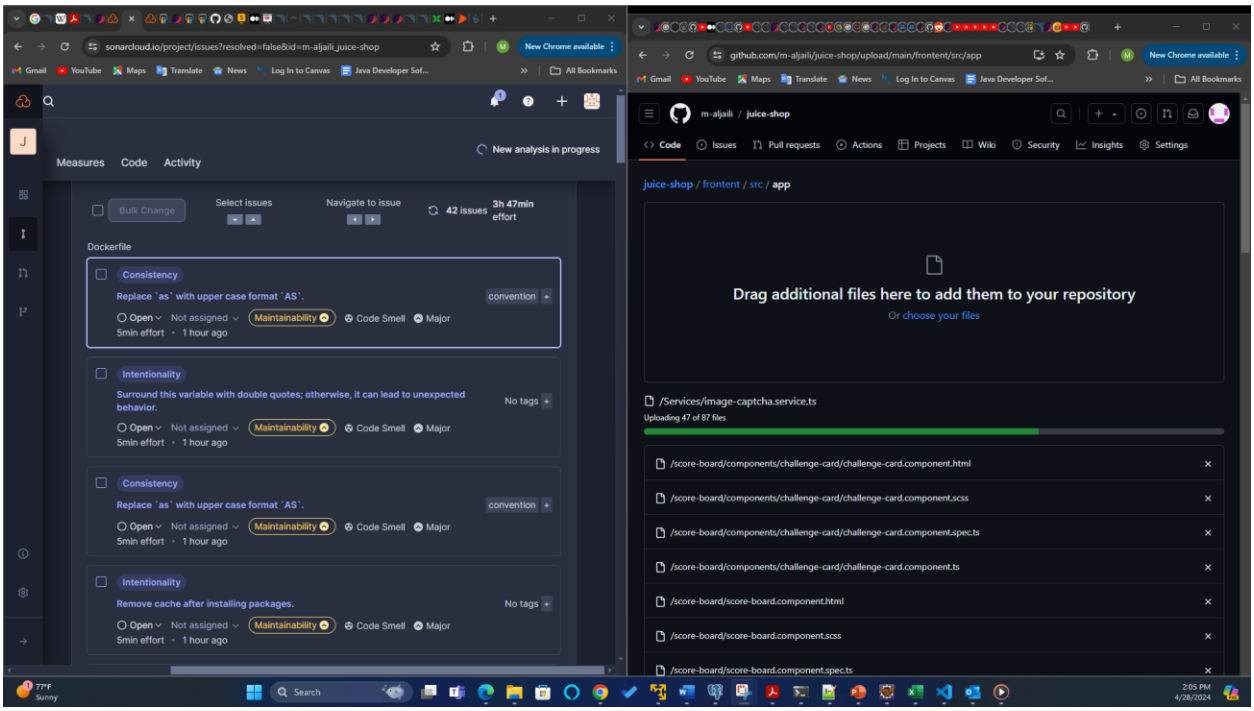

Figure 17: Simultaneous Review while developing new code.

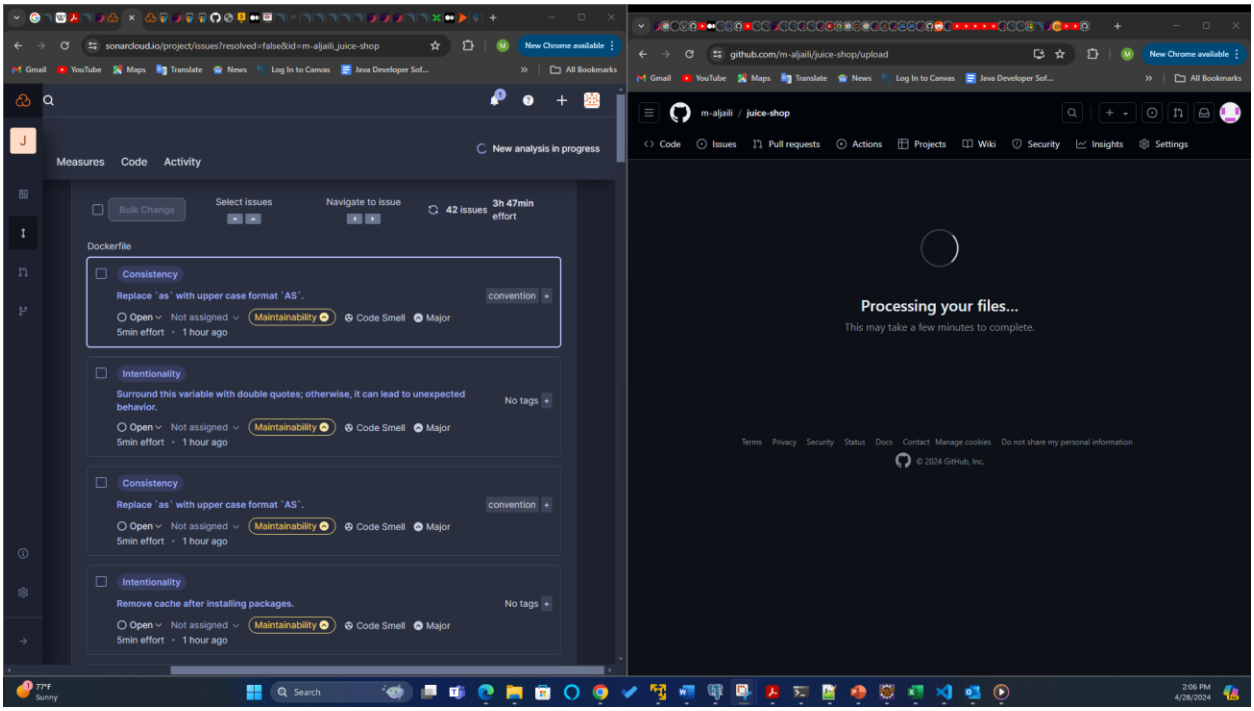

Figure 18: Showing Simultaneous updates on review report.

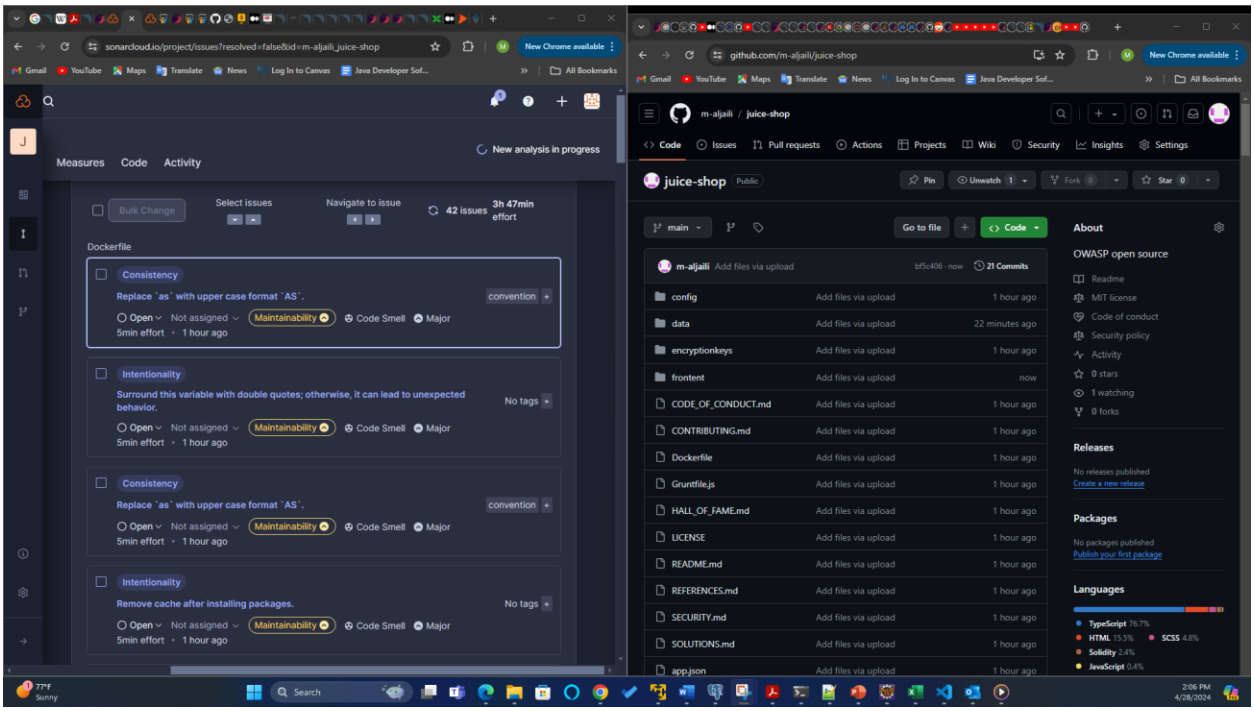

Figure 19: New issues reported while coding.

Conclusion

SonarQube stands out as an indispensable tool in the realm of code quality assurance. By facilitating continuous inspection and automated analysis, it empowers developers to identify and rectify issues promptly, ensuring that the codebase not only meets but exceeds quality standards. Its ability to integrate with various programming languages and CI/CD tools makes it a versatile choice for projects of any scale. Ultimately, SonarQube contributes to a more maintainable, efficient, and secure codebase, which is crucial in today's fast-paced software development environment. Adopting SonarQube can lead to significant improvements in code quality, consistency, and overall project health, making it a wise investment for any development team aiming for excellence.

Refrences

1. https://www.sonarsource.com/
2. https://docs.sonarsource.com/sonarqube/latest/