ATTACK DEFENSE

by PentesterAcademy

Name	SonarQube: Continuous Code Quality Monitoring
URL	https://www.attackdefense.com/challengedetails?cid=2052
Type	DevSecOps Basics: Static Code Analysis

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

SonarQube is an open-source platform used to continuously monitor the code quality of the application by performing static code analysis.

A Kali CLI machine (kali-cli) and a SonarQube server (sonar) are provided in the lab. The SonarQube scanner client is installed on the Kali machine that will scan the web application and push the results to the SonarQube server machine where it can be accessed by the user in the form of reports.

The credentials for accessing the report from SonarQube server interface are:

Objective: Use SonarQube to perform the static code analysis on the application and find issues!

Instructions:

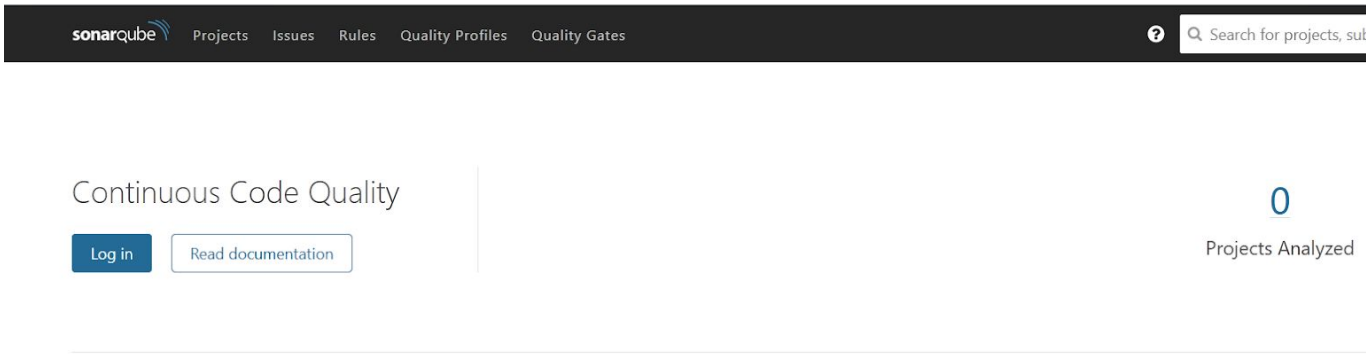
- The sonarqube server can be reached at “sonar” hostname in the lab.

Lab Setup

On starting the lab, the following interface will be accessible to the user.

On choosing (clicking the text in the center) top left panel, **KALI CLI** will open in a new tab

Similarly on selecting the top right panel, a web UI of **SonarQube UI** will open in a new tab.



Solution

Step 1: Check the available options of sonar-scanner.

Command: sonar-scanner --help

```
root@kali-gui:~# sonar-scanner --help
INFO:
INFO: usage: sonar-scanner [options]
INFO:
INFO: Options:
INFO:  -D,--define <arg>     Define property
INFO:  -h,--help             Display help information
INFO:  -v,--version          Display version information
INFO:  -X,--debug            Produce execution debug output
root@kali-gui:~#
```

Step 2: Check the provided source code of web applications.

Command: ls -l github-repos/

```
root@kali-gui:~# ls -l github-repos/
total 12
drwxr-xr-x 7 root root 4096 Sep 13 12:12 django-todolist
drwxr-xr-x 4 root root 4096 Sep 13 12:12 java-mvn-hello-world-web-app
drwxr-xr-x 6 root root 4096 Sep 13 12:12 sonarqube-scanner-gradle
root@kali-gui:~#
```

We will take one example at a time and run the tool on that.

Example 1: Django Todolist

Step 1: Change to the django-todolist directory and check its contents.

Commands:
cd github-repos/django-todolist/
ls

```
root@kali-gui:~# cd github-repos/django-todolist/
root@kali-gui:~/github-repos/django-todolist#
root@kali-gui:~/github-repos/django-todolist# ls
accounts  LICENSE  manage.py  README.pa         sonar-project.properties
api       lists    README.md  requirements.txt  todolist
root@kali-gui:~/github-repos/django-todolist#
```

Step 2: Check the sonar.project.properties file

Command: cat sonar-project.properties

```
root@kali-gui:~/github-repos/django-todolist# cat sonar-project.properties
sonar.projectKey=django-project
sonar.host.url=http://sonar:9000
sonar.sources=.
root@kali-gui:~/github-repos/django-todolist#
```

Explanation:

- **projectKey:** Project Name or Name of the application
- **Host URL:** The URL of sonar server
- **Sources:** Path to directories containing main files.

Step 3: Run the sonar-scanner command to start the sonarqube scan.

Command: sonar-scanner

```
root@kali-gui:~/github-repos/django-todolist# sonar-scanner
INFO: Scanner configuration file: /opt/sonar-scanner-4.4.0.2170/conf/sonar-scanner.properties
INFO: Project root configuration file: /root/github-repos/django-todolist/sonar-project.properties
INFO: SonarScanner 4.4.0.2170
INFO: Java 11.0.5 Debian (64-bit)
INFO: Linux 4.15.0-72-generic amd64
INFO: User cache: /root/.sonar/cache
INFO: Scanner configuration file: /opt/sonar-scanner-4.4.0.2170/conf/sonar-scanner.properties
INFO: Project root configuration file: /root/github-repos/django-todolist/sonar-project.properties
INFO: Analyzing on SonarQube server 7.5.0
```

```
INFO: Default locale: "en_US", source code encoding: "UTF-8" (analysis is platform dependent)
INFO: Publish mode
INFO: Load global settings
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.protobuf.UnsafeUtil (file:/root/.sonar/cache/193d1645c91fbb0
7781506b7df9db0b9/sonar-scanner-engine-shaded-7.5-all.jar) to field java.nio.Buffer.address
WARNING: Please consider reporting this to the maintainers of com.google.protobuf.UnsafeUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

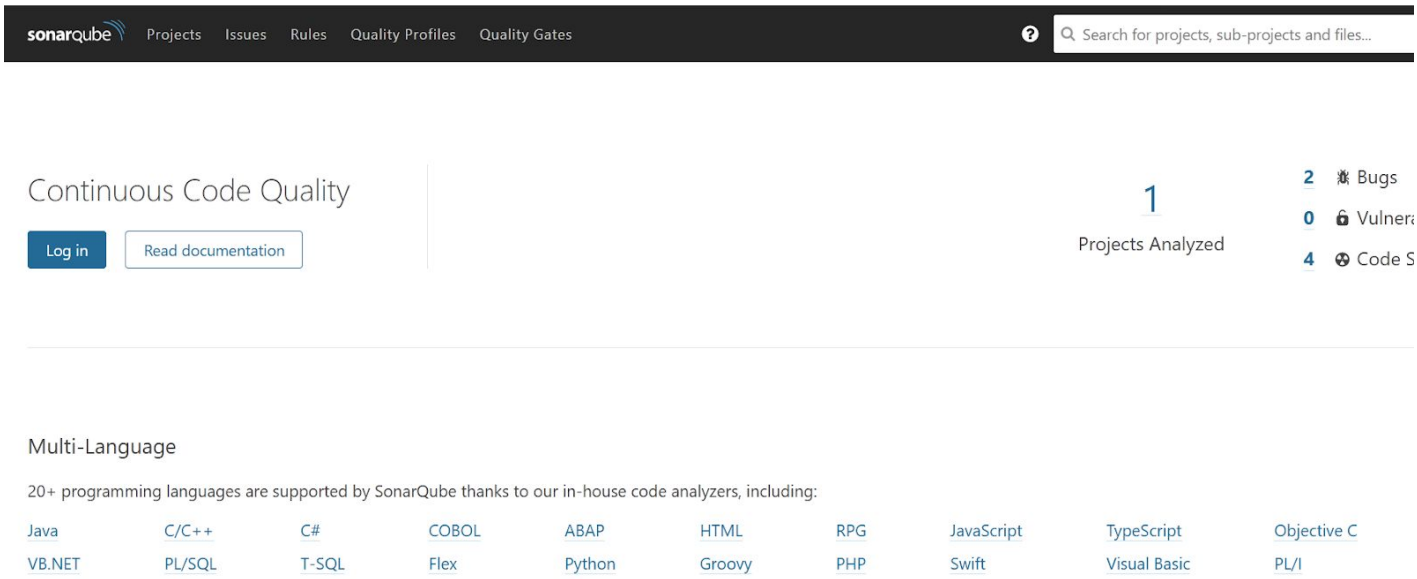
```
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=52ms
INFO: No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
INFO: 15 files had no CPD blocks
INFO: Calculating CPD for 17 files
INFO: CPD calculation finished
INFO: Analysis report generated in 186ms, dir size=171 KB
INFO: Analysis reports compressed in 87ms, zip size=71 KB
INFO: Analysis report uploaded in 340ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://sonar:9000/dashboard?id=django-project
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
INFO: More about the report processing at http://sonar:9000/api/ce/task?id=AXStmIEf9ChHS_lXMrTR
INFO: Task total time: 14.569 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 17.623s
INFO: Final Memory: 13M/47M
INFO: ------------------------------------------------------------------------
root@kali-gui:~/github-repos/django-todolist#
```

The sonar-scanner has generated a report which has been uploaded to the sonarqube server.

Step 4: Navigate to the SonarQube server and login using the credentials provided in the challenge description.

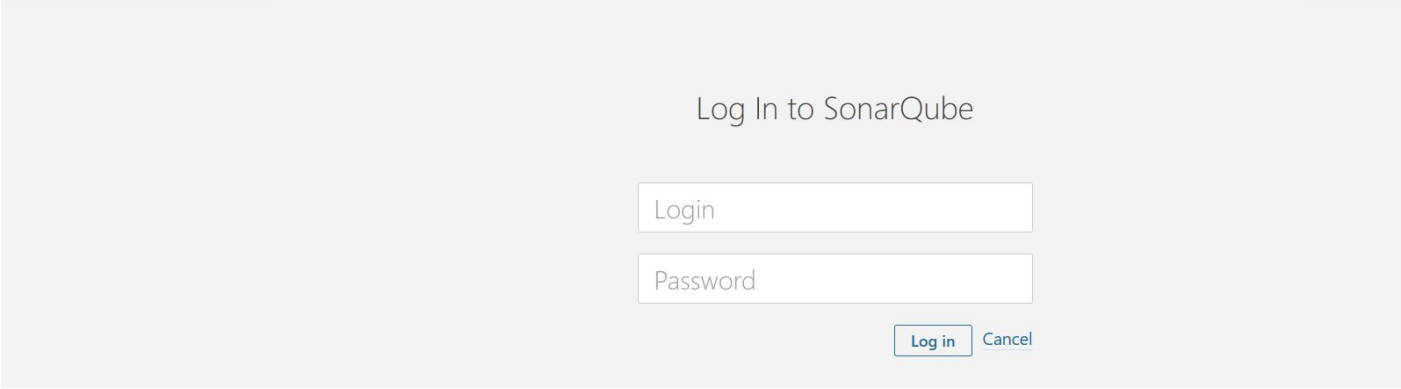
Credentials:

- **Username:** admin
- **Password:** admin

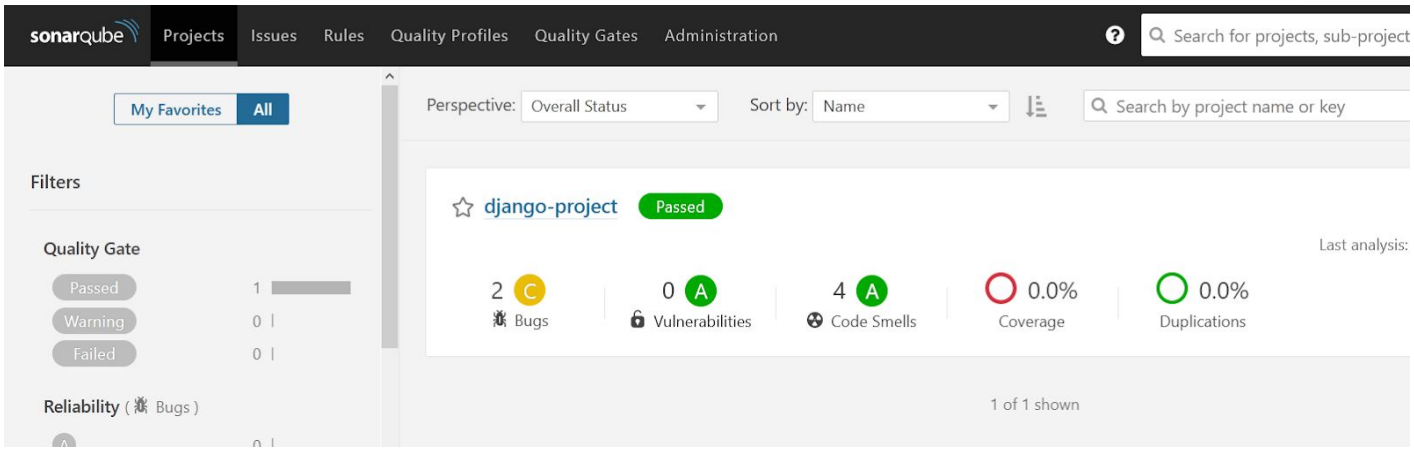


Click on the login button.

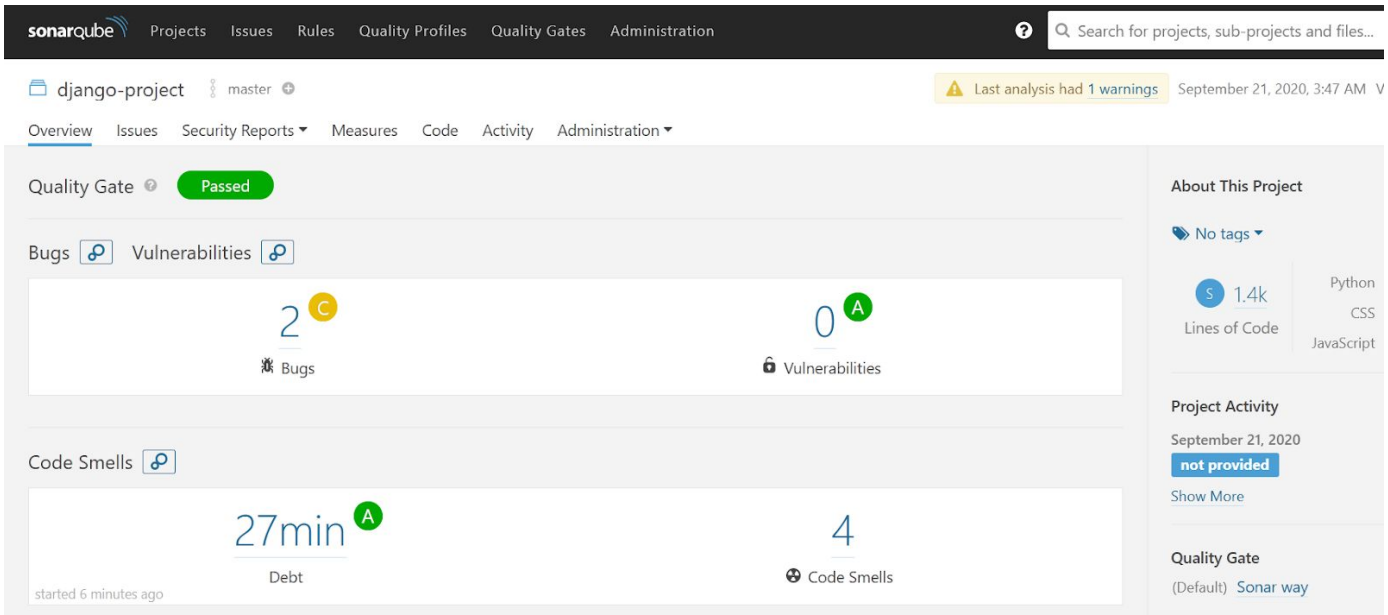
Login Panel:



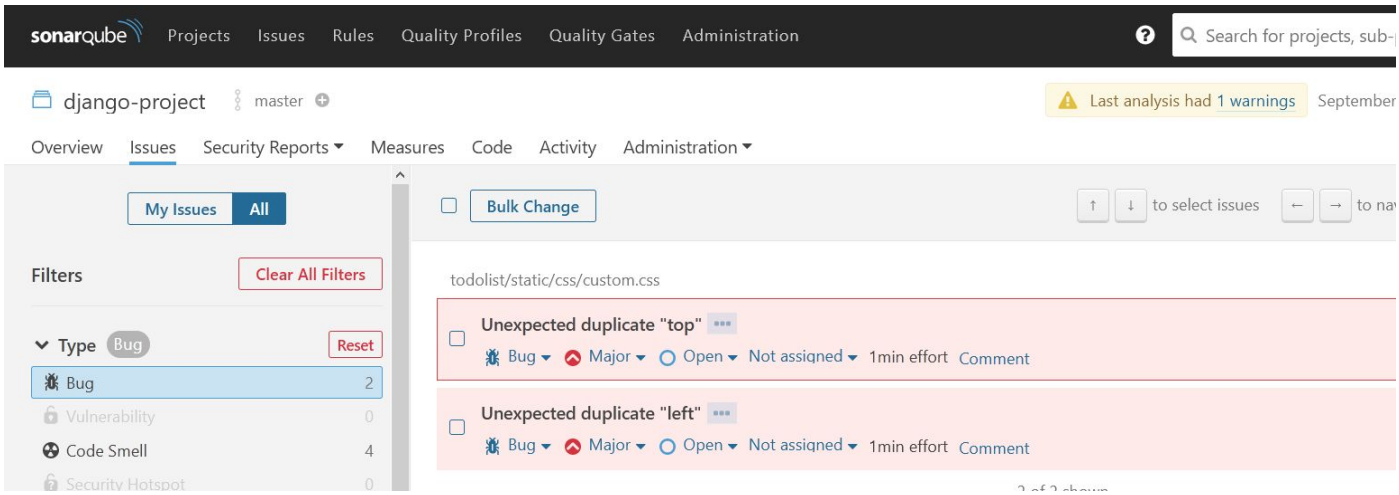
Admin Dashboard:



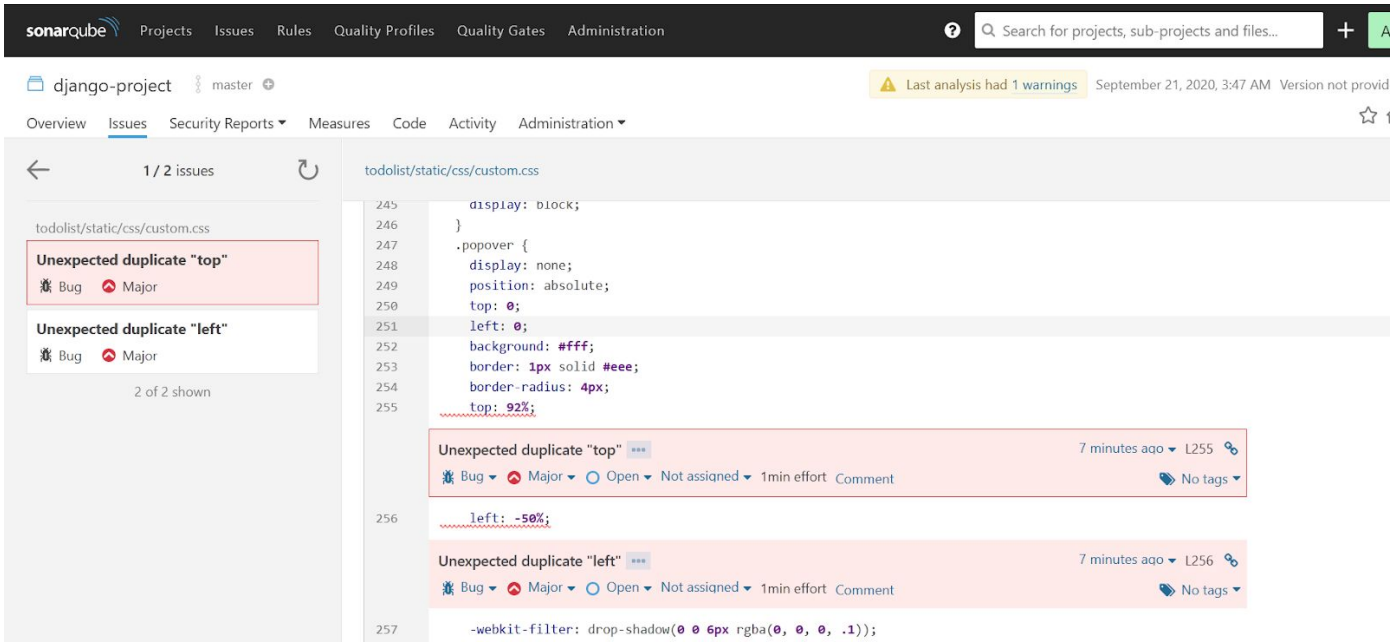
Step 6: Click on the django-project to get the summary of the application.



Click on the Bugs button to get the list of bugs found in the application.



Step 7: Click on one of the bugs to get the details where the bug is present in the code



Issues Detected

- Duplicate entries found in the custom.css of the application

Example 2: SonarQube Scanner Gradle Project

Step 1: Change to the sonarqube-scanner-gradle directory and check its contents.

Commands:
cd ~/github-repos/sonarqube-scanner-gradle
ls

```
root@kali-gui:~/github-repos# cd sonarqube-scanner-gradle/
root@kali-gui:~/github-repos/sonarqube-scanner-gradle#
root@kali-gui:~/github-repos/sonarqube-scanner-gradle# ls
build         gradle             gradlew      README.md  settings.gradle
build.gradle  gradle.properties  gradlew.bat  README.pa  src
root@kali-gui:~/github-repos/sonarqube-scanner-gradle#
```

Step 2: In this case, the SonarQube is integrated into the build process, so it will automatically execute during build. The following configuration file ensures that integration.

Command: cat build.gradle

```
root@kali-gui:~/github-repos/sonarqube-scanner-gradle# cat build.gradle
plugins {
    id "jacoco"
    id "java"
    id "application"
    id "org.sonarqube" version "3.0"
}

description = 'Example of SonarQube Scanner for Gradle Usage'
version = '1.0'

sonarqube {
    properties {
        property 'sonar.projectName', 'Example of SonarQube Scanner for Gradle Usage'
    }
}
```

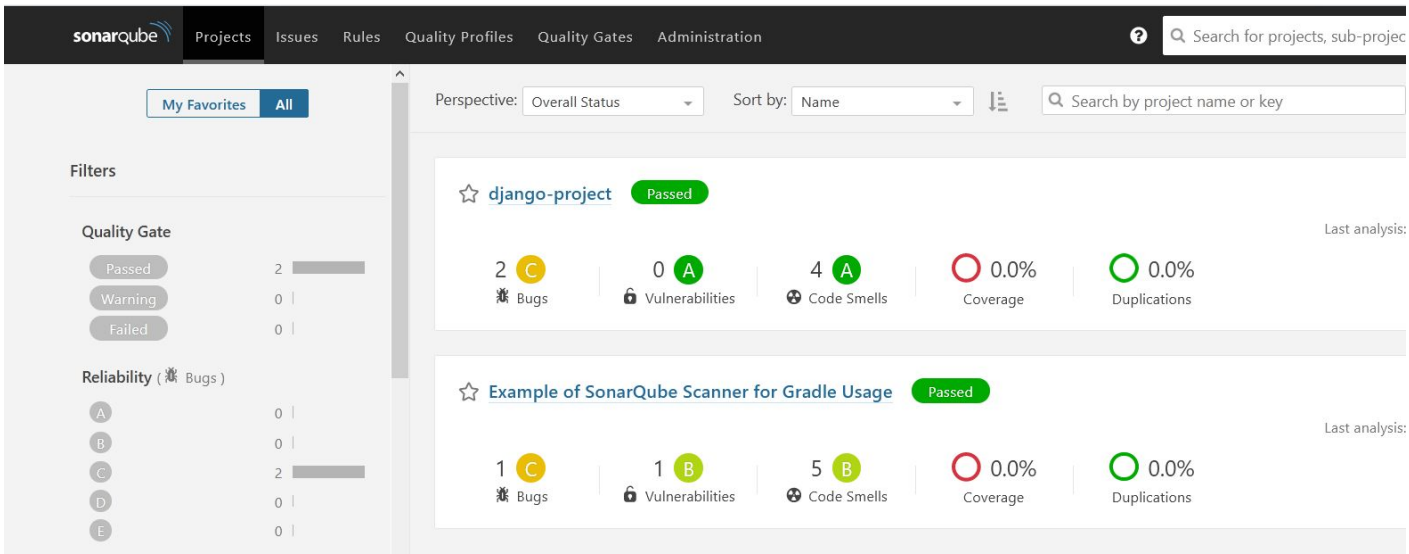
Run the gradlew (gradle wrapper) with the sonar URL to scan the application using the sonarqube.

Command: ./gradlew -Dsonar.host.url=http://sonar:9000 sonarqube

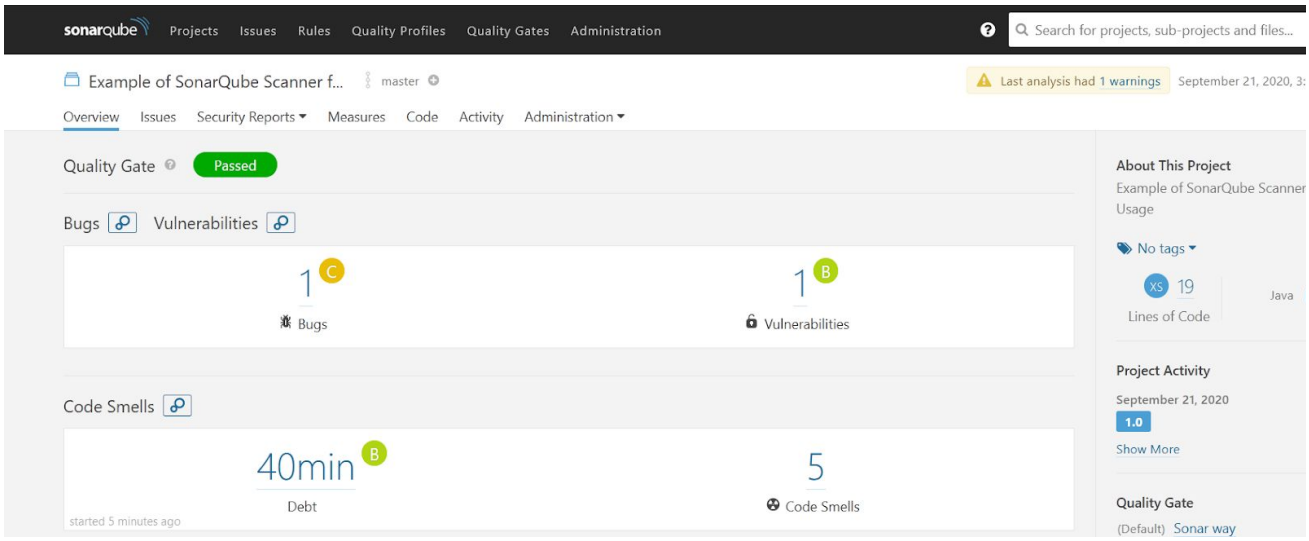
```
root@kali-gui:~/github-repos/sonarqube-scanner-gradle# ./gradlew -Dsonar.host.url=http://sonar:9000 sonarqube
Starting a Gradle Daemon, 2 incompatible and 1 stopped Daemons could not be reused, use --status for details

> Task :sonarqube
SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable
the SCM Sensor in the project settings.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.4.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 18s
3 actionable tasks: 3 executed
root@kali-gui:~/github-repos/sonarqube-scanner-gradle#
```

The project has been built as well as the report is generated on the sonarqube server.

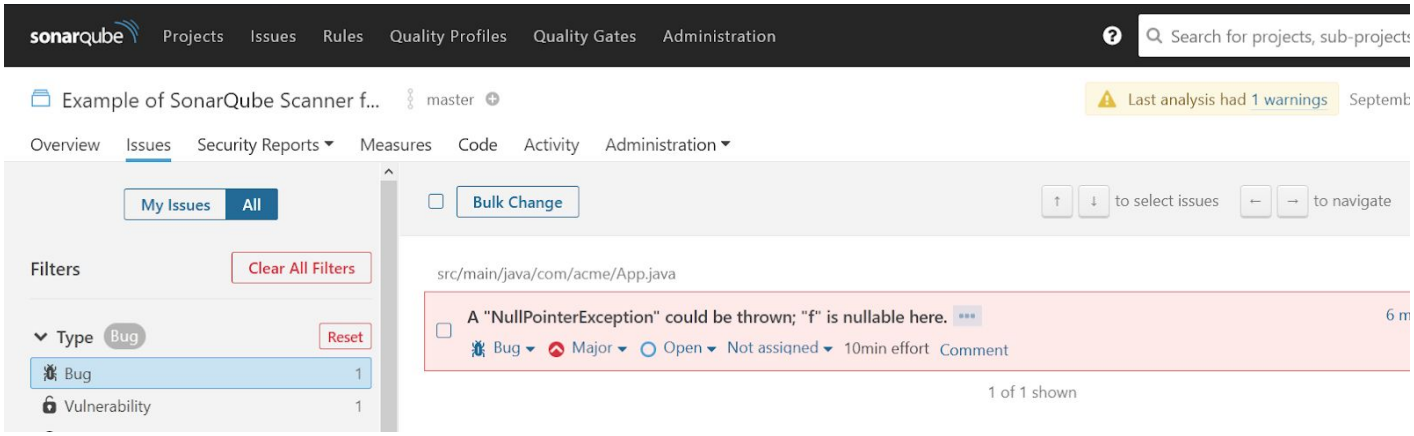
Step 3: Navigate to the sonarqube server to check the results.



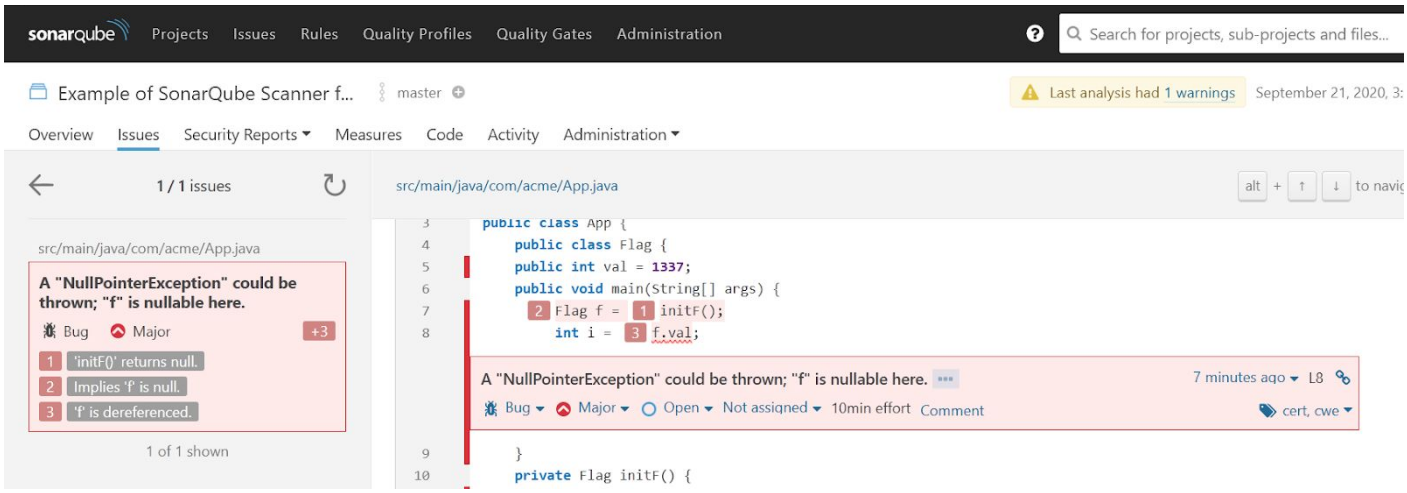
The project has been added in the sonarqube, click on the “Example of SonarQube Scanner for Gradle Usage”



Step 4: Click on the “Bugs” button to check the vulnerabilities present in the application.



Click on the vulnerability found.



Issues Detected

- NullPointerException on variable f in App.java file.

Example 3: Java Maven Hello World Web App

Step 1: Change to the java-mvn-hello-world-web-app directory and check its contents.

Commands:
cd ~/github-repos/java-mvn-hello-world-web-app
ls

```
root@kali-gui:~/github-repos# cd java-mvn-hello-world-web-app/
root@kali-gui:~/github-repos/java-mvn-hello-world-web-app#
root@kali-gui:~/github-repos/java-mvn-hello-world-web-app# ls
ApplicationManifest.yml  LICENSE  README.md  sample_jenkins_file   sonar-project.properties  target
Jenkinsfile              pom.xml  README.pa  SecurityManifest.yml  src
root@kali-gui:~/github-repos/java-mvn-hello-world-web-app#
```

Step 2: In this case, the SonarQube is integrated in the build process, so it will automatically execute during build. The following configuration file ensures that integration.

Command: cat pom.xml

```
<plugin>
        <groupId>org.sonarsource.scanner.maven</groupId>
        <artifactId>sonar-maven-plugin</artifactId>
        <version>3.4.0.905</version>
</plugin>
```

Build the Java project using maven as well as scan the application using sonarqube.

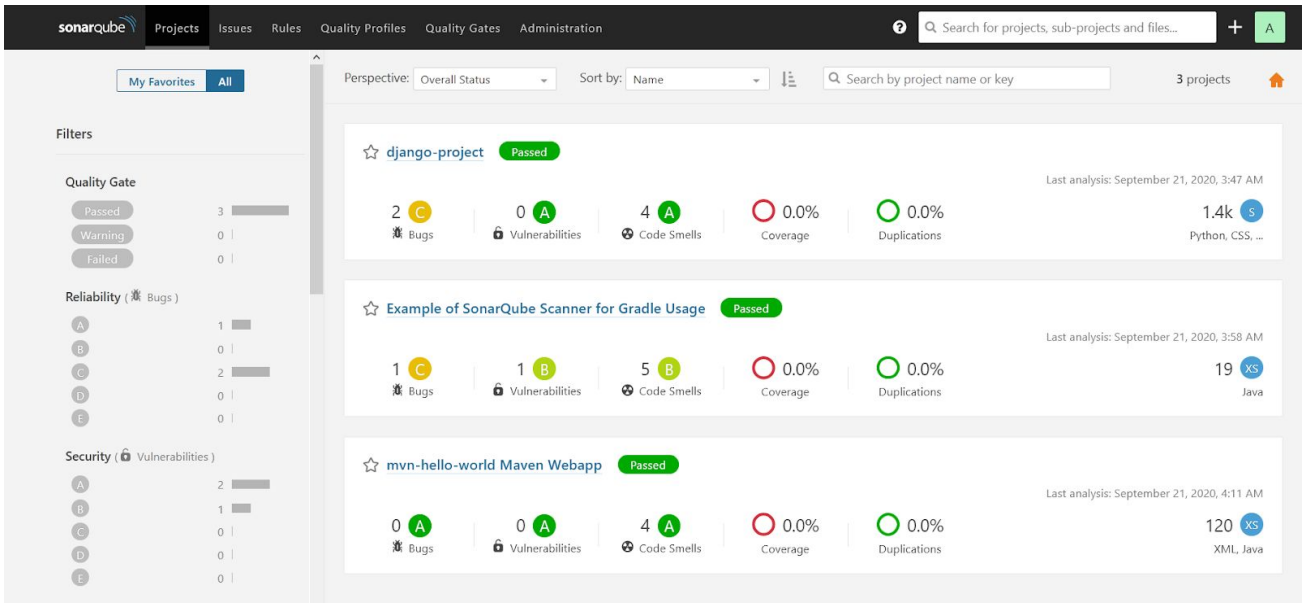
Command: mvn sonar:sonar -Dsonar.host.url=http://sonar:9000

```
root@kali-gui:~/github-repos/java-mvn-hello-world-web-app# mvn sonar:sonar -Dsonar.host.url=http://sonar:9000
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------< com.dev3l.hello_world:mvn-hello-world >----------------
[INFO] Building mvn-hello-world Maven Webapp 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- sonar-maven-plugin:3.4.0.905:sonar (default-cli) @ mvn-hello-world ---
[INFO] User cache: /root/.sonar/cache
[INFO] SonarQube version: 7.5.0
[INFO] Default locale: "en_US", source code encoding: "UTF-8" (analysis is platform dependent)
[INFO] Publish mode
[INFO] Load global settings
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.protobuf.UnsafeUtil (file:/root/.sonar/cache/193d1645c91fbb0
7781506b7df9db0b9/sonar-scanner-engine-shaded-7.5-all.jar) to field java.nio.Buffer.address
WARNING: Please consider reporting this to the maintainers of com.google.protobuf.UnsafeUtil
```

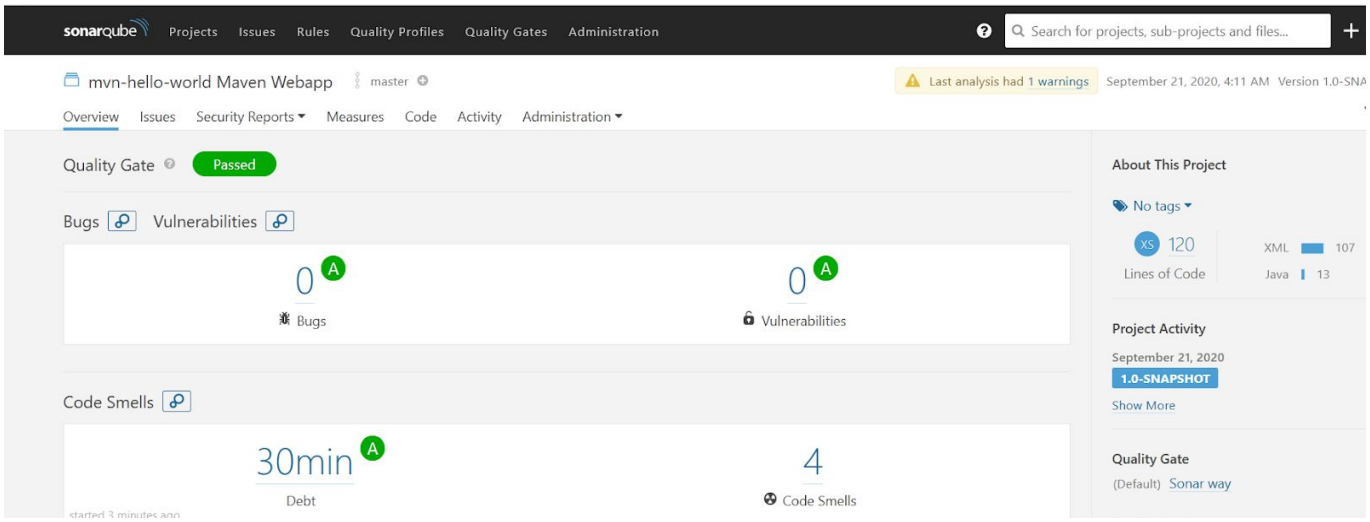
```
[INFO] Analysis report uploaded in 224ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://sonar:9000/dashboard?id=com.dev3l.hello_world%3Amvn-hello-w
orld
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted
 analysis report
[INFO] More about the report processing at http://sonar:9000/api/ce/task?id=AXStrgHL9ChHS_lXMrTg
[INFO] Task total time: 8.917 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.785 s
[INFO] Finished at: 2020-09-20T22:41:36Z
[INFO] ------------------------------------------------------------------------
root@kali-gui:~/github-repos/java-mvn-hello-world-web-app#
```

The project build is completed as well as the scan from sonarqube.

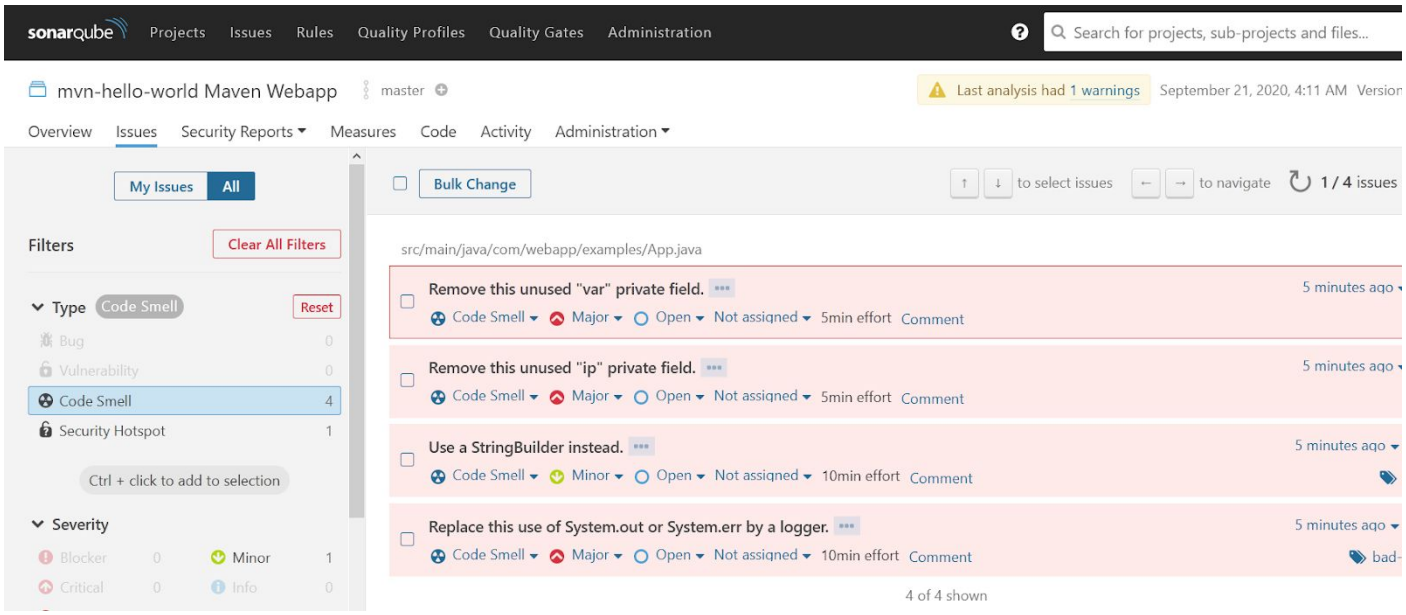
Step 3: Navigate to the sonarqube dashboard



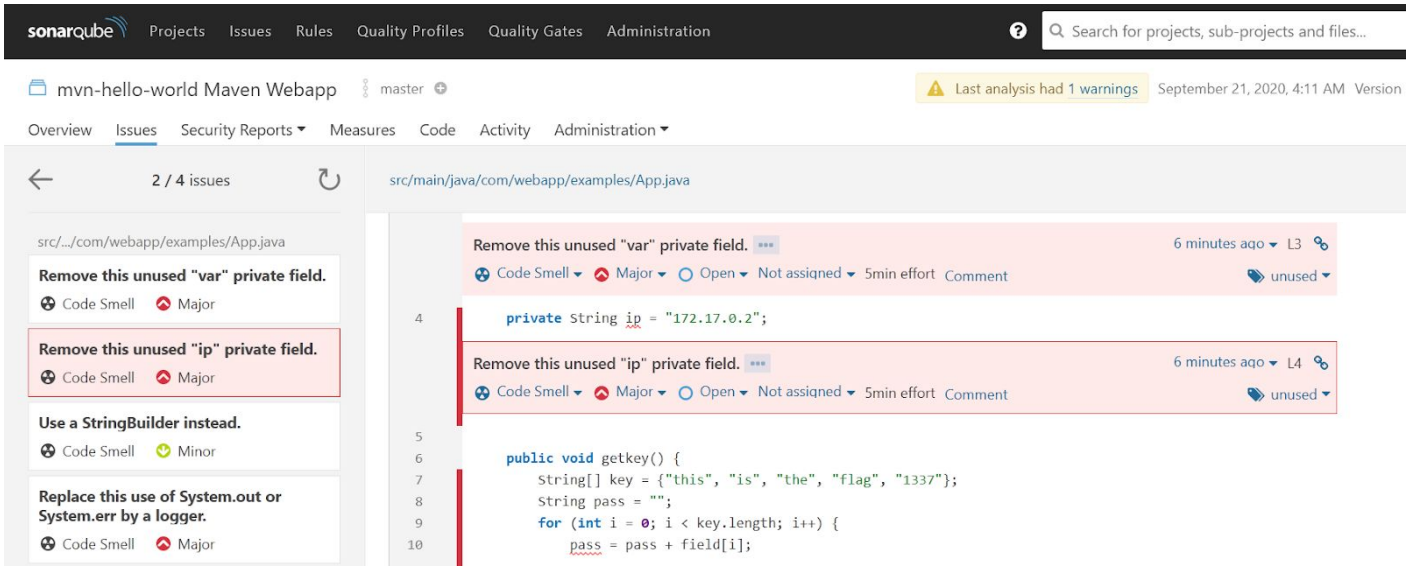
The application has no bugs but 4 coding mistakes, click on the “maven-hello-world Maven Webapp” to get more information



Step 4: Click on the Code Smells section.



Check one of the fields to get more information on the mistakes.



Issues Detected

- Multiple Unused fields found (ip, var)
- StringBuilder should be used
- System.err should be used instead of System.out by a logger.

Learnings

Perform static code analysis with SonarQube tool.