Thibaut SAUTEREAU Follow
Dec 15, 2017 · 9 min read

Using SonarQube to Analyze a Java Project

For the past weeks, I have tried to leverage SonarQube in order to do static analysis of the James source code. This article reports and describes some of the things I did to eventually spot a few security issues in James.

What Is SonarQube?

SonarQube is an open-source platform for continuous inspection of code quality. Using static code analysis, it tries to detect bugs, code smells and security vulnerabilities. SonarQube supports many languages through built-in rulesets and can also be extended with various plugins.

In this article, we are particularly interested in security issues. Many static analysis tools exist for the Java language, including free and open-source ones. Some advantages of SonarQube are the following:

- It is actively developed and well integrated. Many plugins are available to use it as part of continuous integration pipelines, including for Maven, Jenkins and GitHub.
- Its built-in rulesets can be extended with plugins that are more security-oriented. For instance, we will use the FindBugs plugin to take advantage of FindBugs rules.
- It can also report things such as duplicated code, code coverage or coding standards.

Installation

To ease things up, here is the Docker Compose YAML file we are going to use:

```
version: '3'
services:
  mysql:
    image: mysql
    container_name: "mysql-thibaut"
    volumes:
      - ./custom-mysql.cnf:/etc/mysql/conf.d/custom-mysql.c
    environment:
      - MYSQL_ROOT_PASSWORD=sonarqube
      - MYSQL_DATABASE=sonarqube
      - MYSQL_USER=sonarqube
      - MYSQL_PASSWORD=sonarqube
  sonarqube:
    image: sonarqube:6.7
    container_name: "sonarqube-thibaut"
    hostname: "sonarqube-thibaut"
    links:
      - mysql
    ports:
      - "9000:9000"
    volumes:
```

Below is the corresponding `start.sh` script that serves as an entry point. It is mainly responsible for creating a new user whose credentials were defined above and with high privileges. We also add the JAR file for the OWASP Dependency Check that we will introduce later.

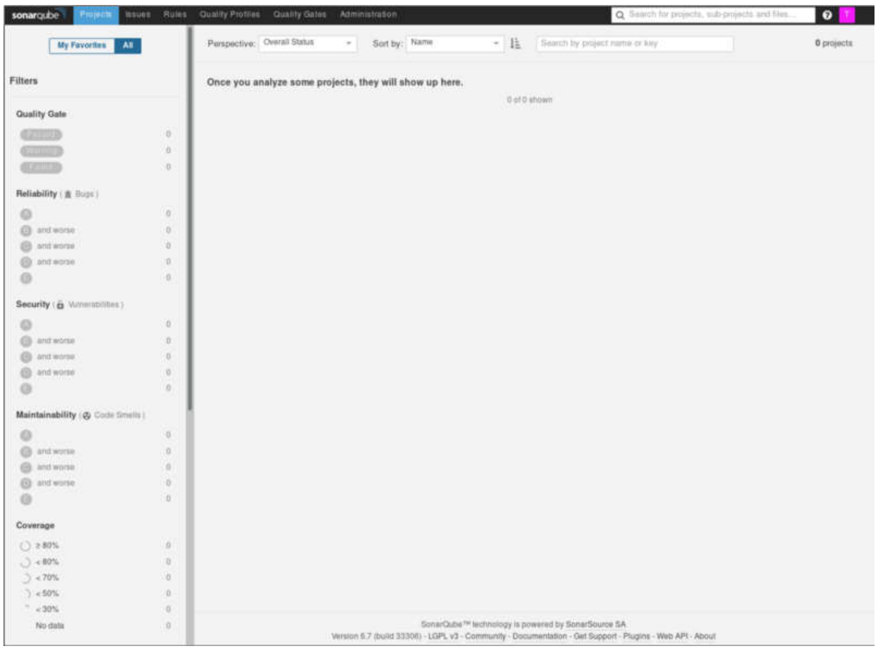
```
#!/bin/bash

# Wait for MySQL database to be running
sleep 20

# Start SonarQube
./bin/run.sh &

curlAdmin() {
    curl -u admin:admin $@
}

URL_API=http://localhost:9000/api

isUp() {
    curl -s -u admin:admin -f "$URL_API/system/info"
}

# Wait for server to be up
while [ -z "$(isUp)" ] ; do sleep 5 ; done

# Provision an admin user
if [ "$USER_LOGIN" ] && [ "$USER_NAME" ] && [ "$USER_PASSWO
then
    curlAdmin "$URL_API/users/create" \
                    --data-urlencode "login=$USER_LOGIN" \
                    --data-urlencode "name=$USER_NAME" \
                    --data-urlencode "password=$USER_PASSWO
    #RIGHTS=( "admin" "profileadmin" ) # Other rights: gate
    #for right in "${RIGHTS[@]}" ; do
```

One should note that we use an external MySQL database instead of the default built-in one. This is important for various reasons, including being able to migrate all the data through SonarQube upgrades. Moreover, here is the additional configuration snippet required to avoid errors later when Maven will be sending analysis reports to SonarQube:

```
[mysqld]
max_allowed_packet = 1073741824
```

As for the OWASP Dependency Check JAR file, you can download it here.

We can now use our web browser to log in to the SonarQube web interface on port `9000` and see the following page:



Once you analyze some projects, they will show up here.

0 of 0 shown

SonarQube™ technology is powered by SonarSource SA
Version 6.7 (build 33306) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

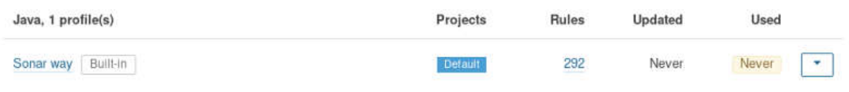
SonarQube startup page after login

As expected, the page is empty and we need to create and setup a new project. Before doing so, we can already take a look at the `Administration` section and start configuring our SonarQube instance. For instance, in the `General Settings -> Security` section, we can enable the `Force user authentication` option. We can also configure webhooks, setup the database cleaner or configure an email account to send notifications. In the `System` tab, we can for instance check the status of the underlying JVM, of the MySQL database or of the Compute and Search engines. In `Administration -> Security` , we can manage user permissions, user groups and create permission templates.

Setting Up a New Project

Our goal is to run a security-oriented analysis of the James source code. Before running an analysis, we need to setup a new project and assign it a **quality profile**. A quality profile is a collection of rules that will be applied during an analysis. There is a default one for each language.

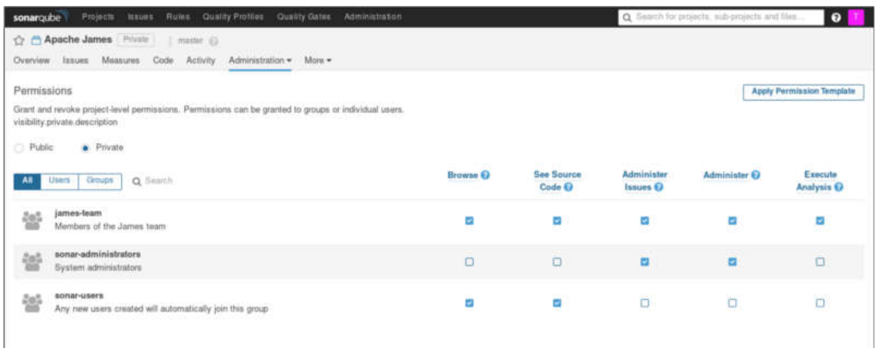
For instance, if we do not do any further configuration and directly analyze the James project, the built-in Java quality profile named *Sonar way* will be used.



Java, 1 profile(s)	Projects	Rules	Updated	Used
Sonar way Built-in	Default	292	Never	Never

The default "Sonar way" built-in Java quality profile

To create the new Apache James project, we go to the `Projects -> Management` section and create a new project. We can choose the name of the project as well as a project key, which needs to be unique. Then, a good practice would be for instance to create a new group with full permissions on this project. We first need to create the group from the general administration panel and then give it rights from the Apache James project administration tab. In our case, we created the `james-team` group for the Apache James project:


sonarqube
Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects, sub-projects and files...
Apache James Private master
Overview Issues Measures Code Activity Administration More
Apply Permission Template

Permissions

Grant and revoke project-level permissions. Permissions can be granted to groups or individual users.
visibility.private.description

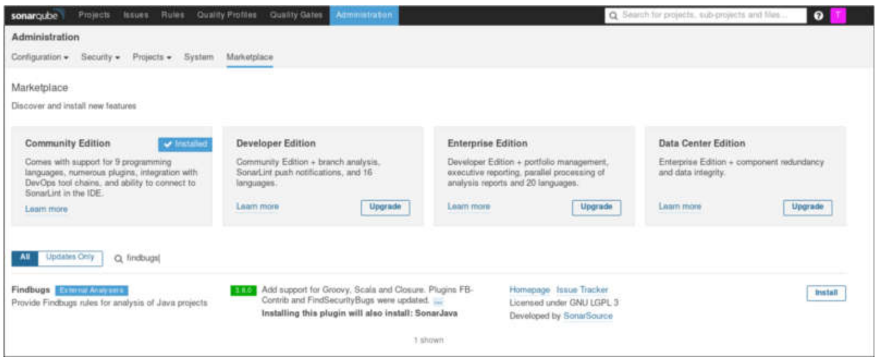
○ Public ● Private

All | Users | Groups Search

	Browse	See Source Code	Administer Issues	Administer	Execute Analysis
james-team Members of the James team	✓	✓	✓	✓	✓
sonar-administrators System administrators	☐	☐	✓	✓	☐
sonar-users Any new users created will automatically join this group	✓	✓	☐	☐	☐

The **james-team** group has full permissions on the Apache James project

We then need to choose a different quality profile for the project. We are going to use the **Findbugs Security Audit** quality profile to benefit from the most security rules available. For that, we need to install the Findbugs plugin. We go to the `Marketplace` tab of the `Administration` section and search for `findbugs` . The first and only result is the good one and we can click on `Install` .



Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Marketplace

Discover and install new features

Community Edition ✓ Installed
Comes with support for 9 programming languages, numerous plugins, integration with DevOps tool chains, and ability to connect to SonarLint in the IDE.
Learn more

Developer Edition
Community Edition + branch analysis, SonarLint push notifications, and 16 languages.
Learn more Upgrade

Enterprise Edition
Developer Edition + portfolio management, executive reporting, parallel processing of analysis reports and 20 languages.
Learn more Upgrade

Data Center Edition
Enterprise Edition + component redundancy and data integrity.
Learn more Upgrade

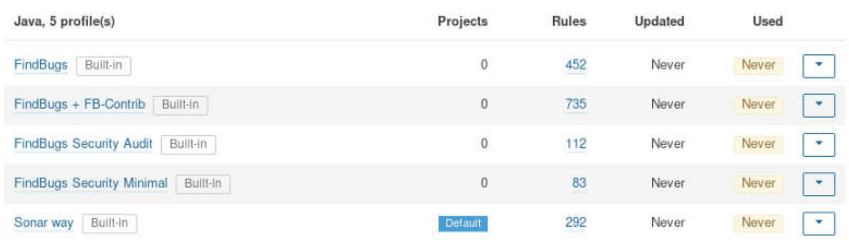
All | Updates Only | findbugs

Findbugs External Analysers
Provide Findbugs rules for analysis of Java projects
3.6.0 Add support for Groovy, Scala and Closure. Plugins FB-Contrib and FindSecurityBugs were updated. ...
Installing this plugin will also install: SonarJava
Homepage Issue Tracker
Licensed under GNU LGPL 3
Developed by SonarSource
Install

1 shown

Let's install the **Findbugs** plugin to gain a lot of new security-oriented rules

We will be asked to restart SonarQube to complete the plugin installation. Once this is done, we notice that new quality profiles are available:

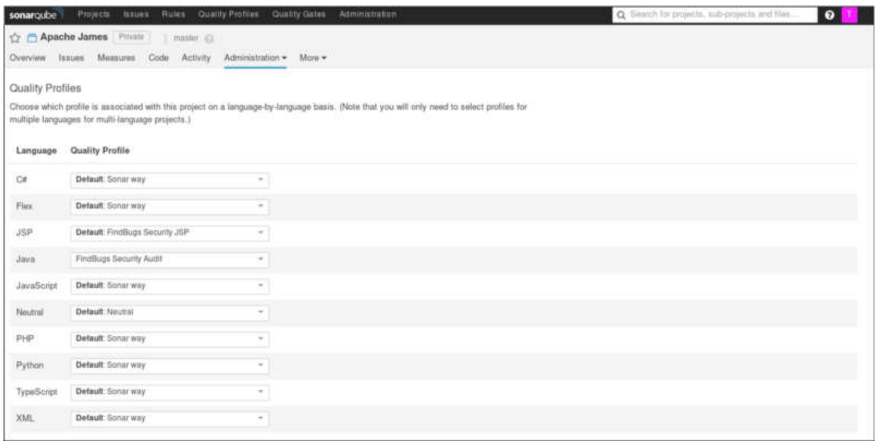


Java, 5 profile(s)	Projects	Rules	Updated	Used
FindBugs Built-in	0	452	Never	Never
FindBugs + FB-Contrib Built-in	0	735	Never	Never
FindBugs Security Audit Built-in	0	112	Never	Never
FindBugs Security Minimal Built-in	0	83	Never	Never
Sonar way Built-in	Default	292	Never	Never

New quality profiles are available thanks to the **Findbugs** plugin

We can check out these new quality profiles and see what rules they import and use. In SonarQube, rules are divided into three self-explaining categories: **bugs**, **vulnerabilities** and **code smells**. By default, some are active and some are not. This can of course be changed.

Back to the Apache James project section, we can now set the quality profile we want to use. As no analysis has been done yet, SonarQube does not know that we are dealing with a Java-only project and thus it is still possible to choose a different quality profile for each language. We obviously limit ourselves to Java and choose the **Findbugs Security Audit** profile.



Quality Profiles

Choose which profile is associated with this project on a language-by-language basis. (Note that you will only need to select profiles for multiple languages for multi-language projects.)

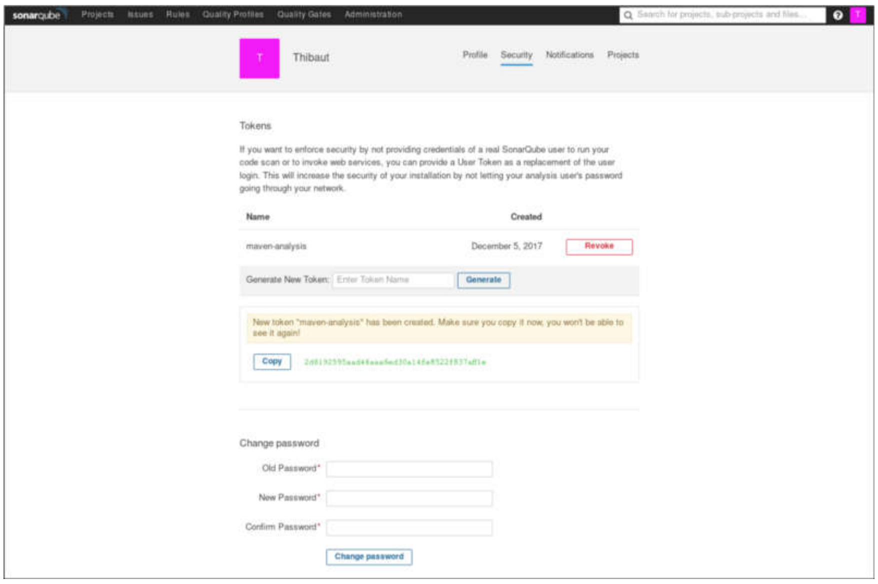
Language	Quality Profile
C#	**Default**: Sonar way
Flex	**Default**: Sonar way
JSP	**Default**: FindBugs Security JSP
Java	FindBugs Security Audit
JavaScript	**Default**: Sonar way
Neutral	**Default**: Neutral
PHP	**Default**: Sonar way
Python	**Default**: Sonar way
TypeScript	**Default**: Sonar way
XML	**Default**: Sonar way

The **Findbugs Security Audit** quality profile is set for the Java language

It is now time to run our first analysis of the Apache James project.

Running an Analysis

James uses Maven as its build automation tool. Maven works well with SonarQube thanks to its SonarQube Scanner plugin. We just need to add a few settings to Maven's configuration and we will then be able to run a simple `mvn` command from the source repository of James. Beforehand, as authentication is required, we generate a security token. For this, we go to the `My Account` section of the user we want to use to run the analysis and we generate a new token:



Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Name	Created	
maven-analysis	December 5, 2017	Revoke

Generate New Token: Enter Token Name [Generate]

New token "maven-analysis" has been created. Make sure you copy it now, you won't be able to see it again!

Copy 2d8192595aad44aaa6ed30a14fa8522f837af1e

Change password

Old Password*

New Password*

Confirm Password*

Change password

A new token is generated and will be given to Maven when running the analysis

Analyzing a Maven project then simply consists of running a Maven goal `sonar:sonar` in the project source directory. We first need to add the following profile to our Maven configuration (usually in `conf/settings.xml`) or in the project's `pom.xml` file:

```
<settings>
  <pluginGroups>
    <pluginGroup>org.sonarsource.scanner.maven</pluginGroup
  </pluginGroups>
  <profiles>
    <profile>
      <id>sonar</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <properties>
        <sonar.host.url>http://myserver:9000</sonar.host.ur
        <sonar.login>2d6192595aad46aaa6ed30a14fe8522f837aff
        <sonar.projectName>Apache James</sonar.projectName>
        <sonar.projectVersion>master</sonar.projectVersion>
```

You can see the security token we generated above. You may also wonder why we are not setting a project key. Actually, Maven automatically uses the following pattern for a project key: **`<groupId>:<artifactId>`** . We cannot do anything about it. If we chose to keep "our" key and set it in the above configuration file, Maven would complain and throw an error. If we used a different key, SonarQube would just create a new different project and we would not benefit from all the settings we have made thus far. Therefore, we must update the key of the project on the SonarQube web interface. In our case, we set it to `org.apache.james:james-project` .

We also exclude all test files from the analysis as well as some other source directories we do not care about.

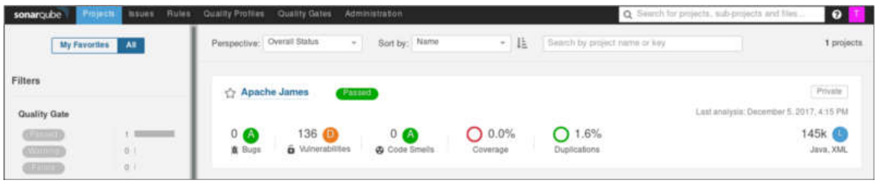
Then, we run the following command to build James and let Maven directly cooperate with our SonarQube server to run the configured quality profile:

```
root@server:~/james-project# mvn clean install sonar:sonar
-DskipTests
```

We also choose to skip the tests to gain time but note that as a consequence SonarQube will not produce any test related data, such as code coverage and test success statistics. Therefore, if you need such things, you should rather run the above command without the `-DskipTests` option.

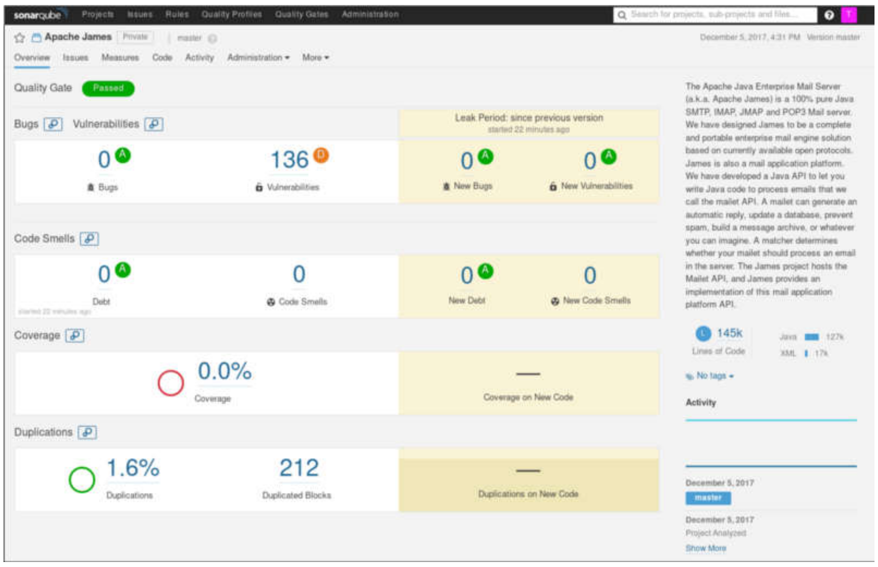
It is worth noting that all the properties added to the XML configuration file above can also be provided as command-line options. This can be useful for instance when such a file is in a Git repository and you do not want to leak the login security token.

Once the build and analysis are completed, you should observe something similar to this in the SonarQube web interface:



Our Apache James project was correctly analyzed

When going to the project's overview page, we see the following:



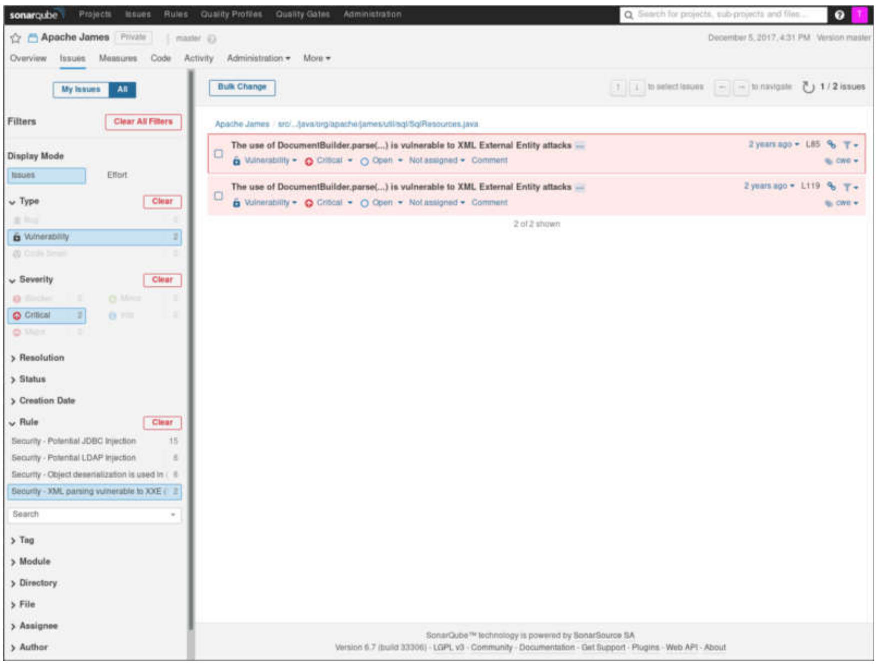
The Apache Java Enterprise Mail Server (a.k.a. Apache James) is a 100% pure Java SMTP, IMAP, JMAP and POP3 Mail server. We have designed James to be a complete and portable enterprise mail engine solution based on currently available open protocols. James is also a mail application platform. We have developed a Java API to let you write Java code to process emails that we call the mailet API. A mailet can generate an automatic reply, update a database, prevent spam, build a message archive, or whatever you can imagine. A matcher determines whether your mailet should process an email in the server. The James project hosts the Mailet API, and James provides an implementation of this mail application platform API.

136 vulnerabilities were reported by SonarQube

We are presented with the number of reported bugs, vulnerabilities and code smells. The code coverage is also displayed but is zero as we did not run the tests. Statistics about code duplication are also showed.

Finally, the evolution since the last analysis is presented so we can quickly see whether the situation has improved or not.

Let's now go to the `Issues` tab. It is possible to use filters on many criteria such as severity, files or rules:



December 5, 2017, 4:31 PM Version master

Security - Potential JDBC Injection 15
Security - Potential LDAP Injection 6
Security - Object deserialization is used in 6
Security - XML parsing vulnerable to XXE 2

Apache James / src/.../java/org/apache/james/util/sql/SqlResources.java

The use of DocumentBuilder.parse(...) is vulnerable to XML External Entity attacks — Vulnerability · Critical · Open · Not assigned · Comment — 2 years ago · L85 · cwe

The use of DocumentBuilder.parse(...) is vulnerable to XML External Entity attacks — Vulnerability · Critical · Open · Not assigned · Comment — 2 years ago · L119 · cwe

2 of 2 shown

SonarQube™ technology is powered by SonarSource SA
Version 6.7 (build 33306) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

We focus on one particular class of reported issues about XML External Entity attacks

We can click on an issue to jump to the related source code. Then we can for instance decide to change the category of the issue, adapt its severity, confirm it or mark it as resolved, assign it to someone, write a comment, etc.

Obviously, many false positives will show up. With other quality profiles that include rules for bugs and code smells, you will probably decide that some do not matter to you. Consequently, you will need to disable certain rules. You may also want to mix rules from different quality profiles. The best way to achieve all of these actions is to create a custom quality profile.

Creating a Custom Quality Profile

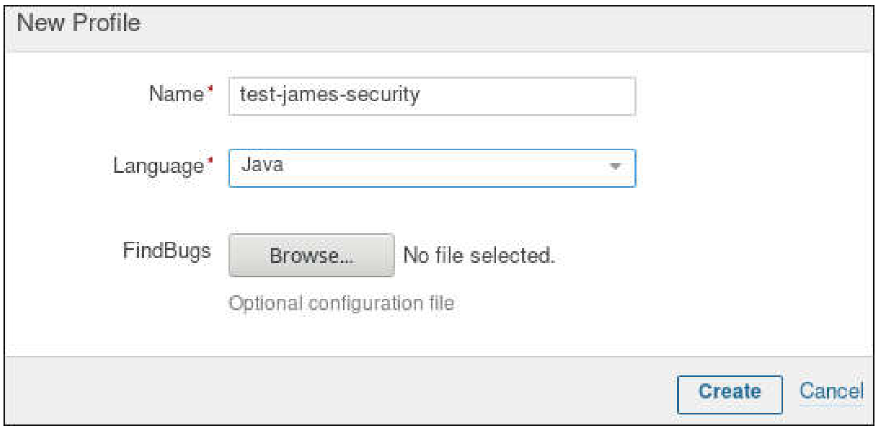
Creating a custom quality profile is interesting because:

> Default profiles cannot be edited, so we will not be able to customize it to our needs.

> That lets us treat default profiles as a baseline against which we can track our own profile as we make changes to it.
>
> Default profiles are typically updated with each new version of their associated plugin to add rules and sometimes adjust rules severities. These changes are automatically applied to existing copies of the profiles.

Note than the following will require us to have the `Administer Quality Profiles` global permission or to be a member of the `sonar-administrators` group, which is already the case for our user thanks to our initial Bash script.

In the `Quality Profiles` section, we create a new quality profile:


New Profile

Name*: test-james-security

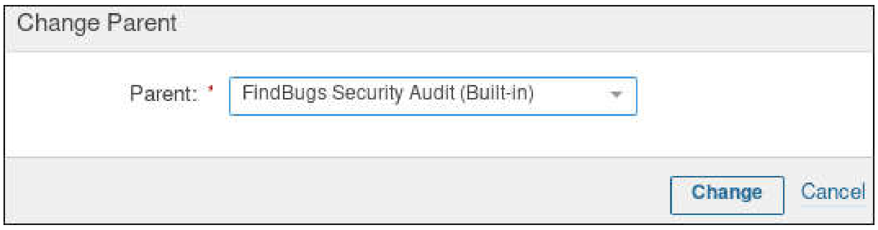
Language*: Java

FindBugs: No file selected.

Optional configuration file

We create a new quality profile named **test-james-security**

We then take advantage of the inheritance between quality profiles by changing the parent of our newly created profile:

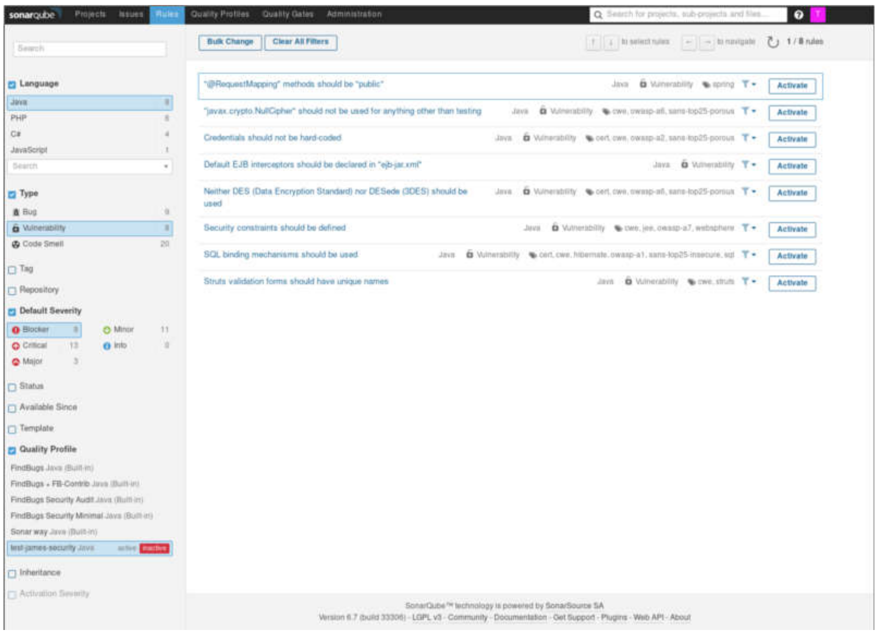

Change Parent

Parent: * FindBugs Security Audit (Built-in)

Our newly created profile will inherit from the **Findbugs Security Audit** profile we used for our first analysis

Do not forget to assign this new quality profile to your project so it will be used for the next analysis.

We can now enable or disable rules. For example, we can look for currently inactive Java rules included in the Findbugs Security Audit quality profile and that are classified as vulnerabilities with a *Blocker* severity:



"@RequestMapping" methods should be "public"	Java	Vulnerability	spring	Activate
"javax.crypto.NullCipher" should not be used for anything other than testing	Java	Vulnerability	cwe, owasp-a6, sans-top25-porous	Activate
Credentials should not be hard-coded	Java	Vulnerability	cert, cwe, owasp-a2, sans-top25-porous	Activate
Default EJB interceptors should be declared in "ejb-jar.xml"	Java	Vulnerability		Activate
Neither DES (Data Encryption Standard) nor DESede (3DES) should be used	Java	Vulnerability	cert, cwe, owasp-a6, sans-top25-porous	Activate
Security constraints should be defined	Java	Vulnerability	cwe, jee, owasp-a7, websphere	Activate
SQL binding mechanisms should be used	Java	Vulnerability	cert, cwe, hibernate, owasp-a1, sans-top25-insecure, sql	Activate
Struts validation forms should have unique names	Java	Vulnerability	cwe, struts	Activate

SonarQube™ technology is powered by SonarSource SA
Version 6.7 (build 33306) - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API - About

We can browse inactive rules and activate some

It is also possible to back up our custom quality profile by downloading it as a file, allowing us to import it later in the same SonarQube instance or in another one. We can also compare different quality profiles, for instance if some day we want to get a summary of how our custom quality profiles have evolved and derived from their original parents.

Finally, note that rules can be flagged as *deprecated*. This merely acts as a warning and does not automatically disable active rules. Deprecations will appear with plugin updates and be propagated to our custom quality profiles since we used inheritance as showed above.

Conclusion

SonarQube is a relatively practical tool and comes with a nice user interface. It can almost work out of the box but one need to spend some time to configure it in order to really take advantage of it. This article was a quick introduction to walk people through a simple setup. There are many facets of SonarQube that we did not cover here. The documentation will help you find out more and we will probably publish another article dealing with more advanced uses of SonarQube (including the interfacing with the OWASP Dependency Check) and summarizing our results.