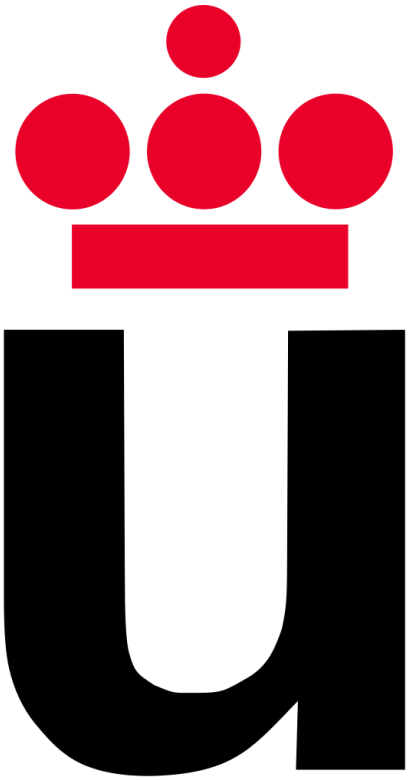
Parte 2 - SonarCloud y un repositorio vulnerable.

Metodologías de desarrollo seguro



Diego Fernández Díaz

Iván Domínguez Romero

Marcos Bonilla Cubero

Pablo Gracia Correa

1. ***¿Cuántas vulnerabilidades, bugs y code smells ha detectado SonarCloud en el proyecto entero?***

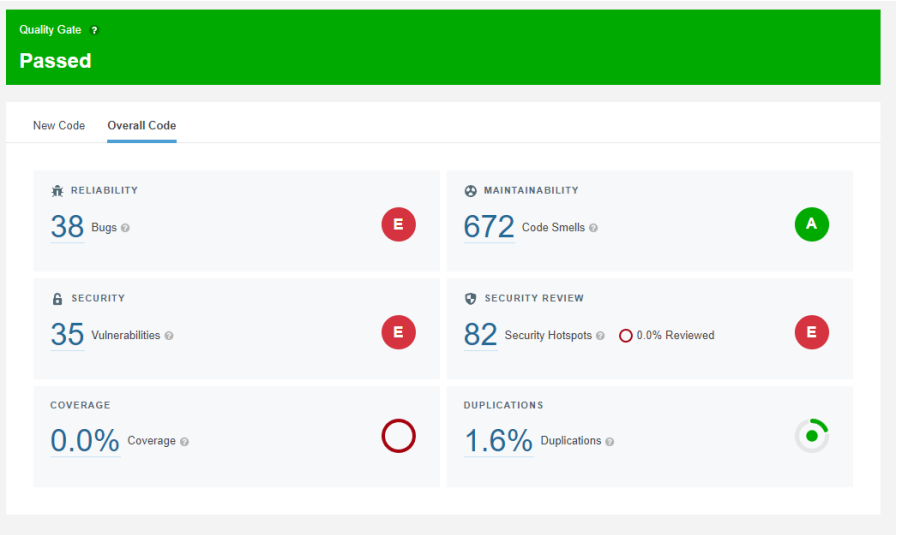
Hemos encontrado un total de 35 vulnerabilidades, 38 bugs y 672 code smells

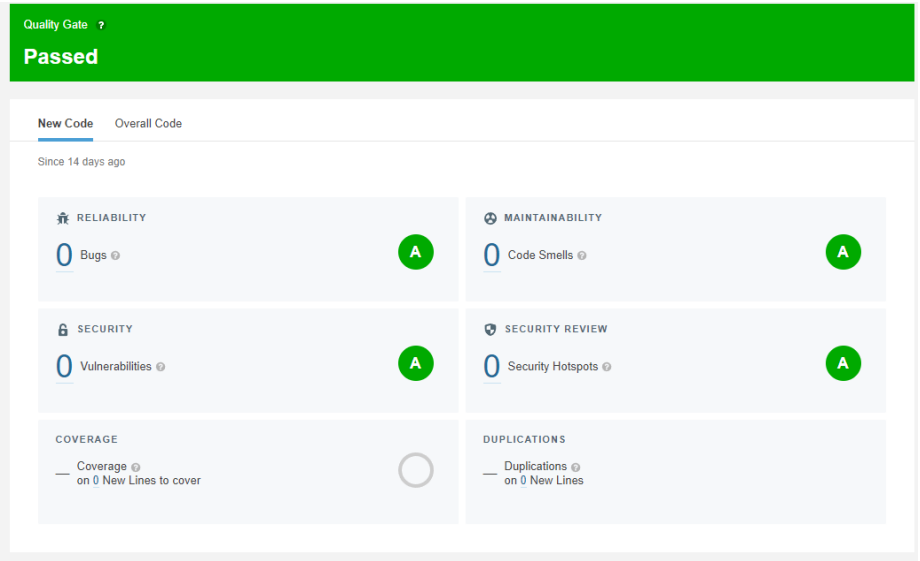
2. ***Haz click sobre el proyecto para abrir la vista de detalle. Revisa las vulnerabilidades detectadas y la lista de Security Hotspots. ¿Qué diferencia crees que existe entre las vulnerabilidades y los Security Hotspots?***

La vulnerabilidad es un agujero de seguridad conocido y confirmado mientras que los security hotspots es una sospecha sobre algo que puede convertirse en una vulnerabilidad y que requiere una revisión manual. Las vulnerabilidades se diferenciarán según su nivel de amenaza(blocker, minor or critical) mientras que los security hotspots se ordenarán por orden de importancia de revisión y llegando a detectar que tipo de ataque podría explotarla.

3. ***Haz cualquier commit para forzar un reanálisis (por ejemplo editando el README.md). Espera a que finalice y vuelve a Sonar. ¿Cuál es el estado del proyecto (Passed/Failed)?. ¿A qué crees que se debe? ¿Crees que el número de vulnerabilidades afecta a dicho veredicto?***

Es passed, el número de vulnerabilidades no ha aumentado ni disminuido, por lo tanto podemos deducir que el resultado es independiente de esto. Consideramos que lo que termina indicando si es passed o failed es la aparición de nuevas vulnerabilidades desde la sincronización con sonar y no la cantidad de las mismas.





1. Elige una vulnerabilidad de tipo Blocker o Critical, explica cual es la vulnerabilidad detectada, porque ha sido detectada (y si realmente es una vulnerabilidad y no un falso positivo).

Hemos resuelto una vulnerabilidad de tipo critical. No se revisaba la firma de un token jwt, la hemos detectado mediante sonarcloud.

2. Propón una solución e impleméntela en una nueva rama del repositorio. Explica los cambios que arreglan dicha vulnerabilidad.

Hemos creado en primer lugar una nueva rama a la que hemos denominado "FIXJWT". Aquí dejamos el hash del comit en nuestro repositorio de github implementando la solución para esta vulnerabilidad "46d88ed93d164cd4b6b0aaa5c54a6c6b820391e4"

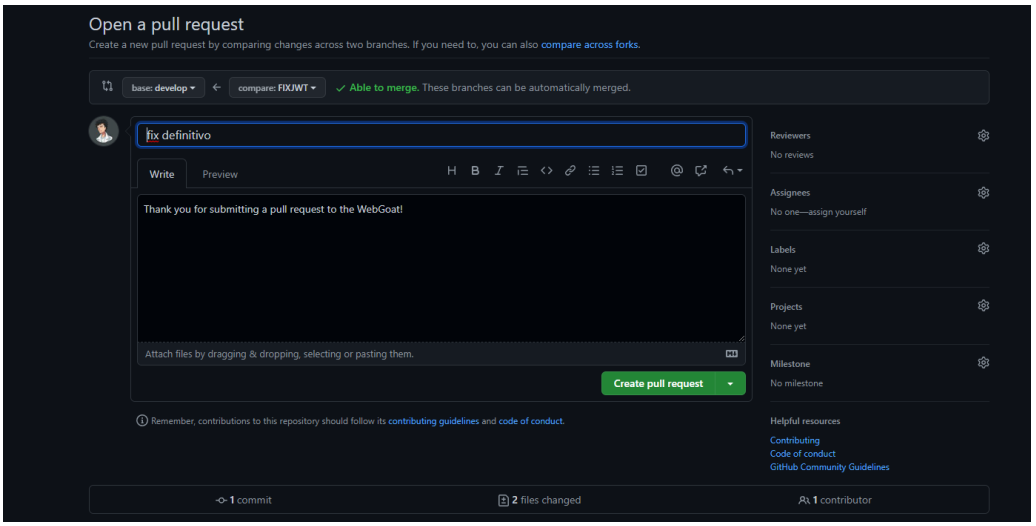
Este era el código original

```
Jwt jwt = Jwts.parser().setSigningKey(JWT_PASSWORD).parse(token.replace("Bearer ",
"")); // vulnerabilidad
```

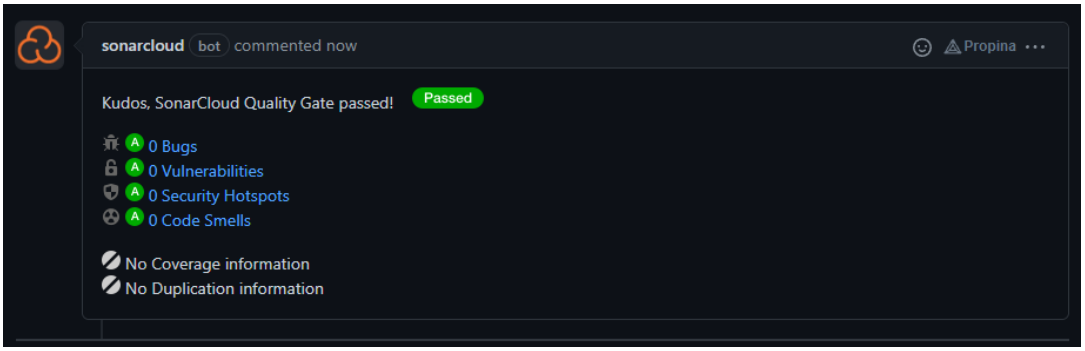
Este es el código resuelto

```
Jwt jwt = Jwts.parser().setSigningKey(JWT_PASSWORD).parseClaimsJws(token.replace("Bearer ", ""));
// vulnerabilidad
```

3. *Crea una Pull Request de la nueva rama a la rama principal. ¿Qué opina Sonar de los cambios realizados?*

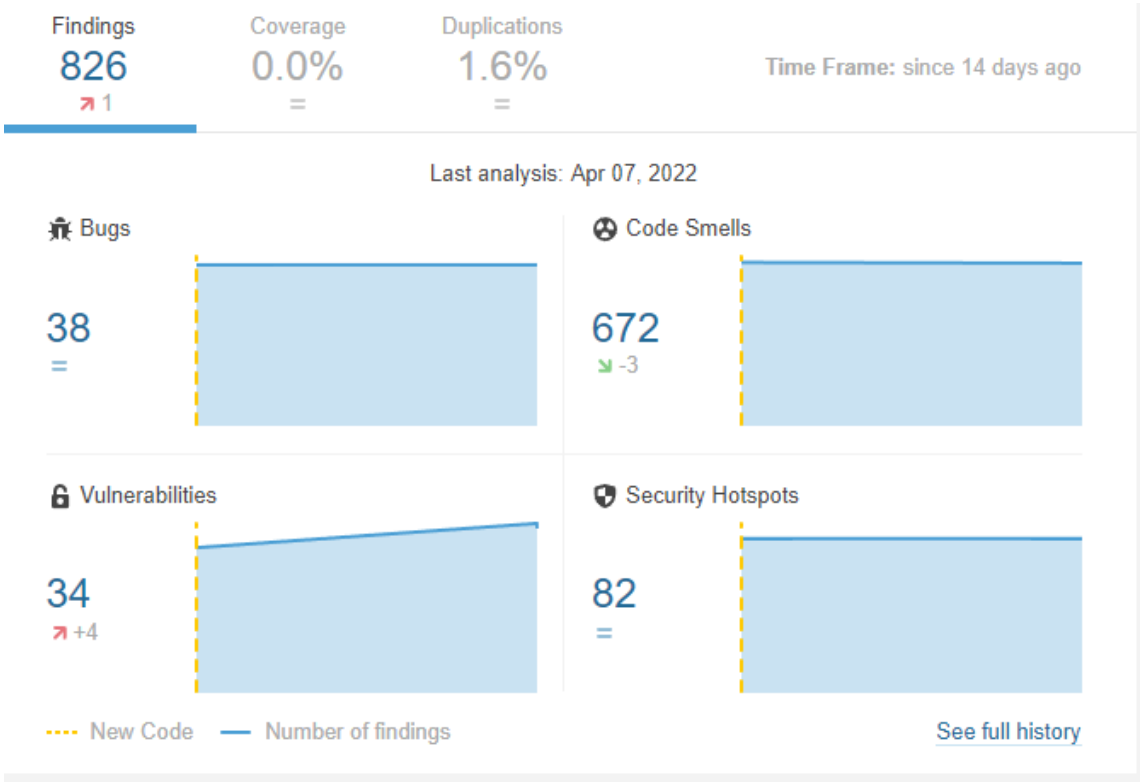


Al terminar de escanear sonarcloud nos ha dejado este comentario en la pull request:



4. *Junta (merge) la nueva rama con la rama principal. Una vez finalizado el análisis, ¿qué cambios se han producido en el proyecto? ¿Cuántas vulnerabilidades detecta ahora?*

Se ha reducido en 1 el número de vulnerabilidades. De forma indirecta también hemos logrado reducir los codesmells en un 3%.



Anotaciones importantes:

Repositorio público parte 2: https://github.com/diegodiaz1256/WebGoat

Repositorio público parte 1:

https://github.com/diegodiaz1256/Practica2Metodologias