

Find Security Bugs

Ejemplos guiados

Realizado por

Grupo 3 – Corocotta

Integrantes

Hamza Hamda

Iván Sánchez Calderón

Juan David Corrales Gil

Ricardo Armando Blanco López

Jaime Eduardo Baires Escalante

Asignatura

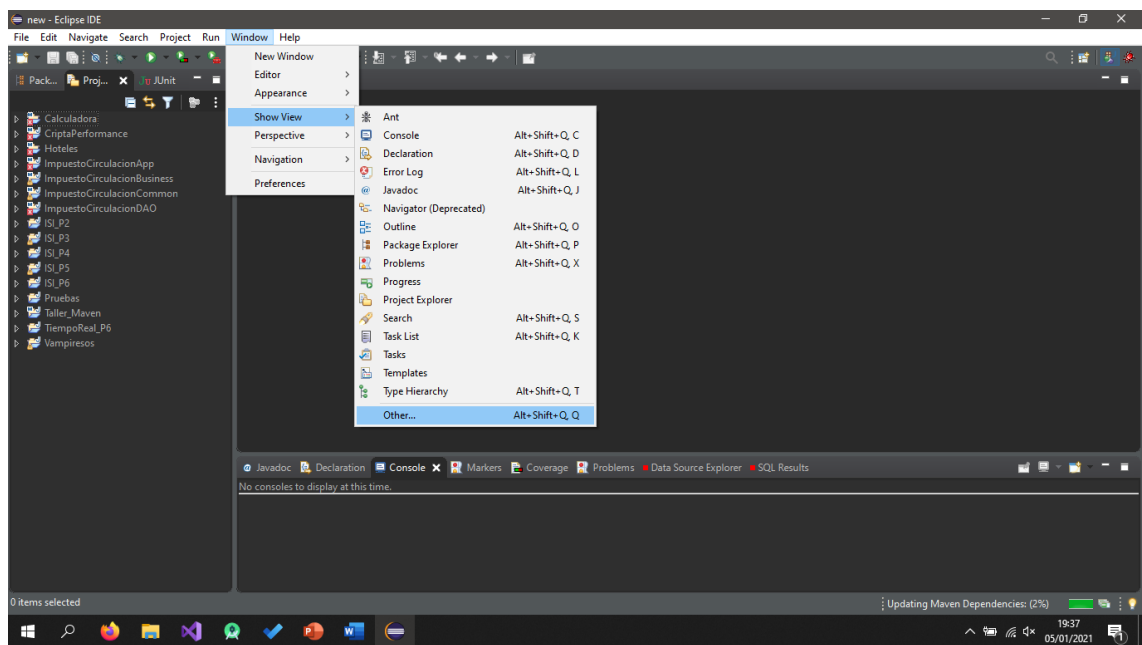
Calidad y Auditoría

Índice

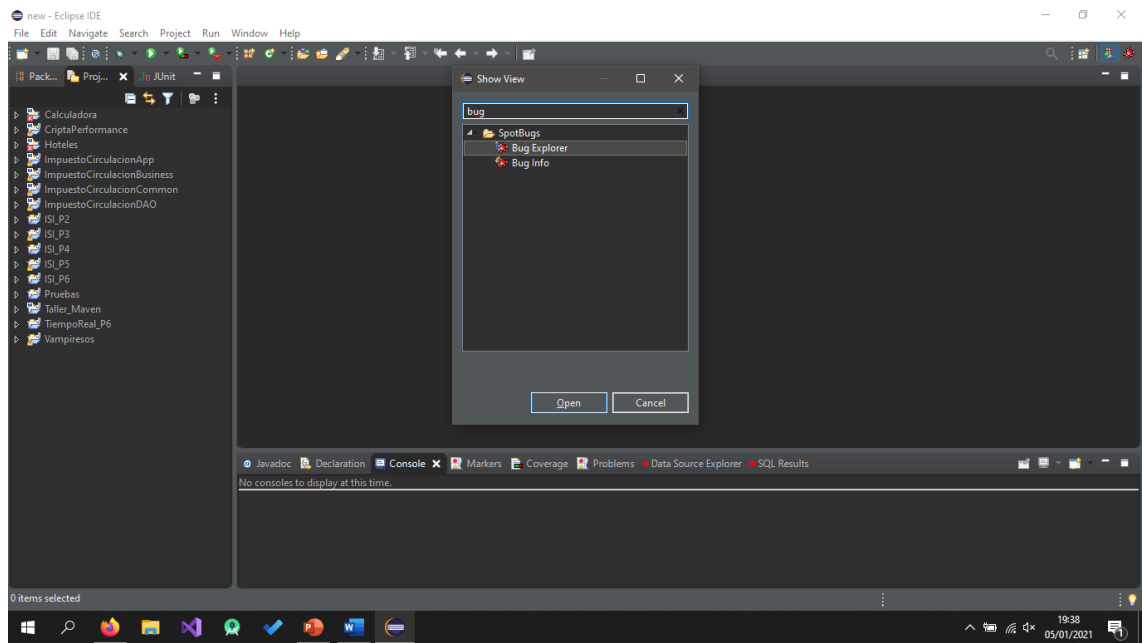
1. Ejemplo guiado con plugin de Eclipse.....	3
2. Ejemplo guiado con plugin de Maven.....	9
3. Ejemplo guiado con SonarQube.....	11

1. Ejemplo guiado con plugin de Eclipse

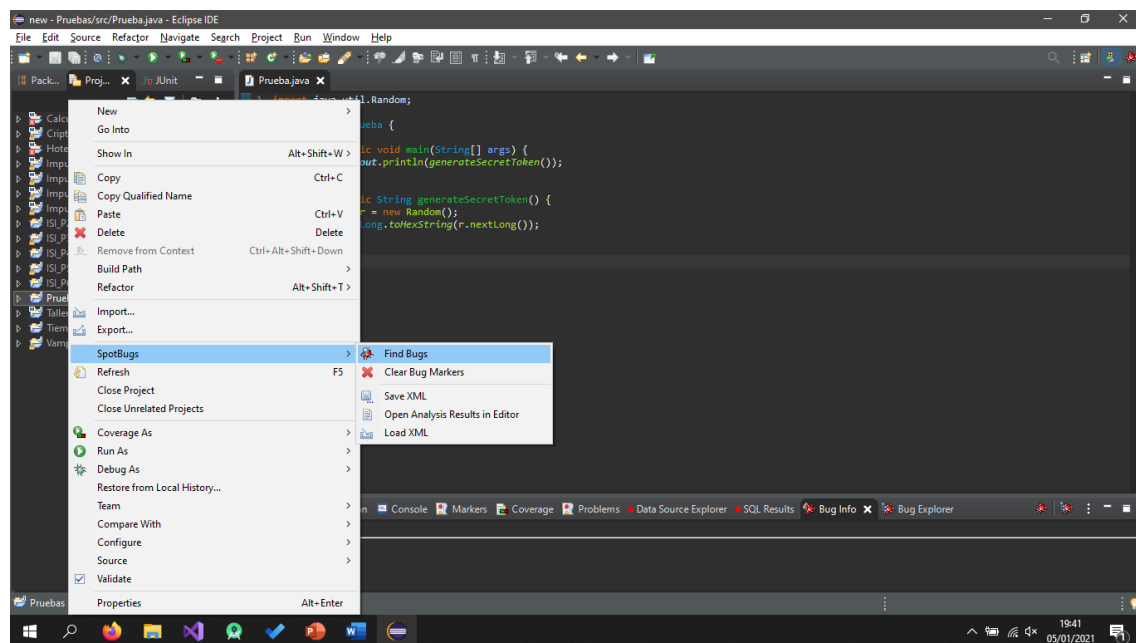
Incluir las vistas relacionadas para poder visualizar los errores. Window -> Show View -> Other...



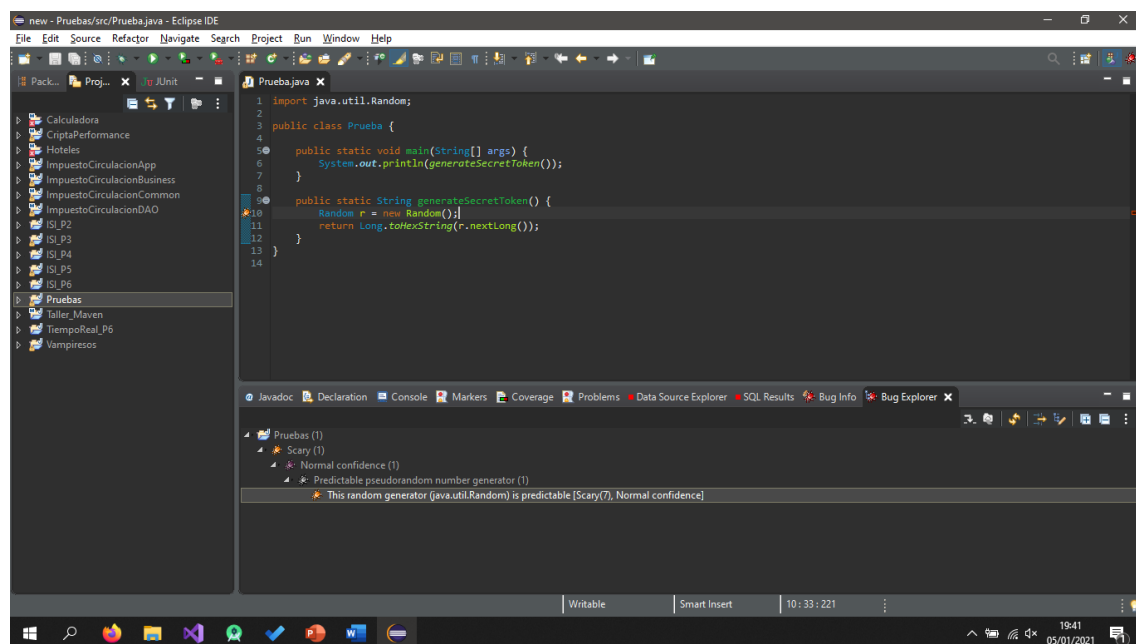
Escribir “bug” y seleccionar las vistas “Bug Explorer” y “Bug Info” -> Open.



Se da clic derecho al proyecto o clase que se quiera analizar -> SpotBugs -> Find Bugs



Al cabo de unos segundos, aparecerían los resultados, si se tienen bugs de seguridad. En la ventana Bug Explorer se encuentra la lista de todos los errores que se tengan en el código, ordenados por categoría.



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The left sidebar displays a project explorer with a tree view of the 'Pruebas' project, showing files like 'Calculadora', 'CargaPerformance', 'Hoteles', 'ImpuestoCirculacionApp', 'ImpuestoCirculacionBusiness', 'ImpuestoCirculacionCommon', 'ImpuestoCirculacionDAO', 'ISL_P2', 'ISL_P3', 'ISL_P4', 'ISL_P5', 'ISL_P6', 'Pruebas', 'Taller_Maven', 'TiempoReal_P6', and 'Vampiresos'. The main editor area shows the 'Prueba.java' file with the following code:

```
1 import java.util.Random;
2
3 public class Prueba {
4
5     public static void main(String[] args) {
6         System.out.println(generateSecretToken());
7     }
8
9     public static String generateSecretToken() {
10         Random r = new Random();
11         return Long.toHexString(r.nextLong());
12     }
13 }
```

Below the code editor, the 'Bug Explorer' tab is active, showing a warning: 'Bug: This random generator (java.util.Random) is predictable'. The warning text states: 'This random generator (java.util.Random) is predictable. Value java.util.Random'. Below this, a text box explains the vulnerability:

Bug: This random generator (java.util.Random) is predictable

The use of a predictable random value can lead to vulnerabilities when used in certain security critical contexts. For example, when the value is used as:

- a CSRF token: a predictable token can lead to a CSRF attack as an attacker will know the value of the token
- a password reset token (sent by email): a predictable password token can lead to an account takeover, since an attacker will guess the URL of the "change password" form
- any other secret value

A quick fix could be to replace the use of java.util.Random with something stronger, such as java.security.SecureRandom.

Vulnerable Code:

```
String generateSecretToken() {
    Random r = new Random();
    return Long.toHexString(r.nextLong());
}
```

The bottom status bar shows the system clock as 19:45 on 05/01/2021.

URL: <https://mvnrepository.com/artifact/commons-codec/commons-codec/1.10>

[Categories](#) | [Popular](#) | [Contact Us](#)

[Indexed Artifacts \(18.7M\)](#)

[Popular Categories](#)

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases

[Home](#) » [commons-codem](#) » [commons-codem](#) » 1.10

Apache Commons Codec » 1.10

The Apache Commons Codec package contains simple encoder and decoders for various formats such as Base64 and Hexadecimal. In addition to these widely used encoders and decoders, the codec package also maintains a collection of phonetic encoding utilities.

License	Apache 2.0
Categories	Base64 Libraries
HomePage	http://commons.apache.org/proper/commons-codec/
Date	(Nov 06, 2014)
Files	pom (11 KB) jar (277 KB) View All
Repositories	Central Apache Releases Redhat GA
Used By	9,450 artifacts

Note: There is a new version for this artifact

New Version	1.15
-------------	------

[Maven](#)
[Gradle](#)
[SBT](#)
[Ivy](#)
[Grape](#)
[Leiningen](#)
[Builder](#)

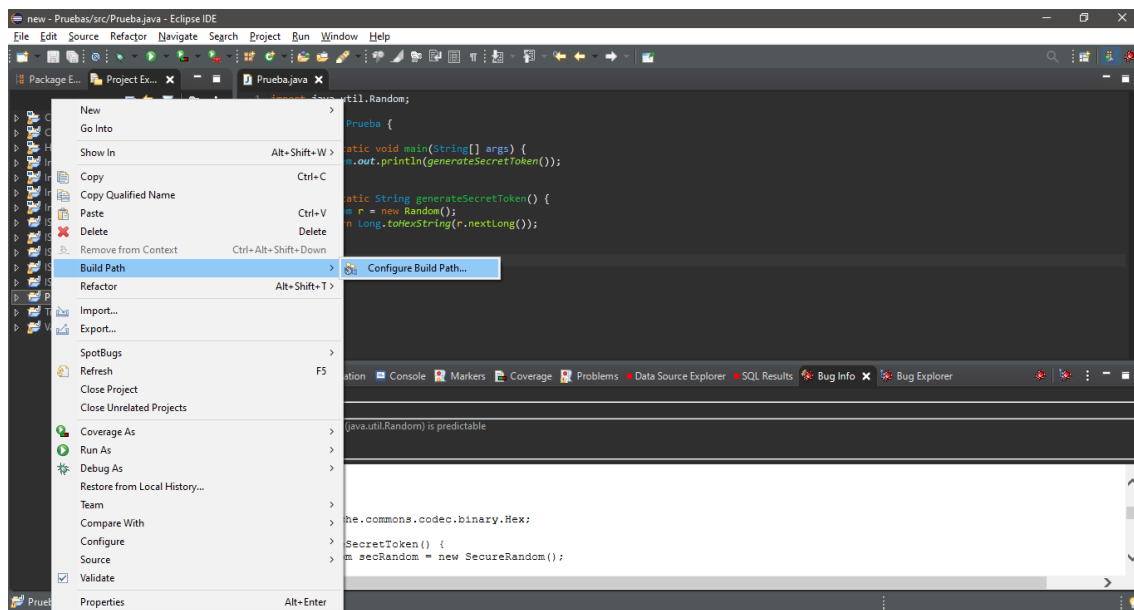
```

<!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.10</version>
</dependency>

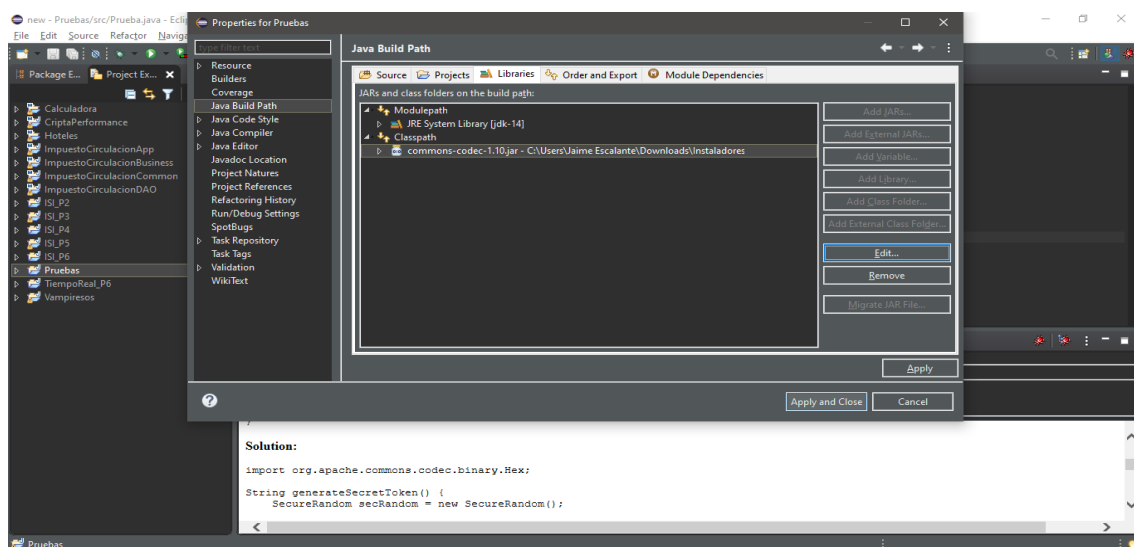
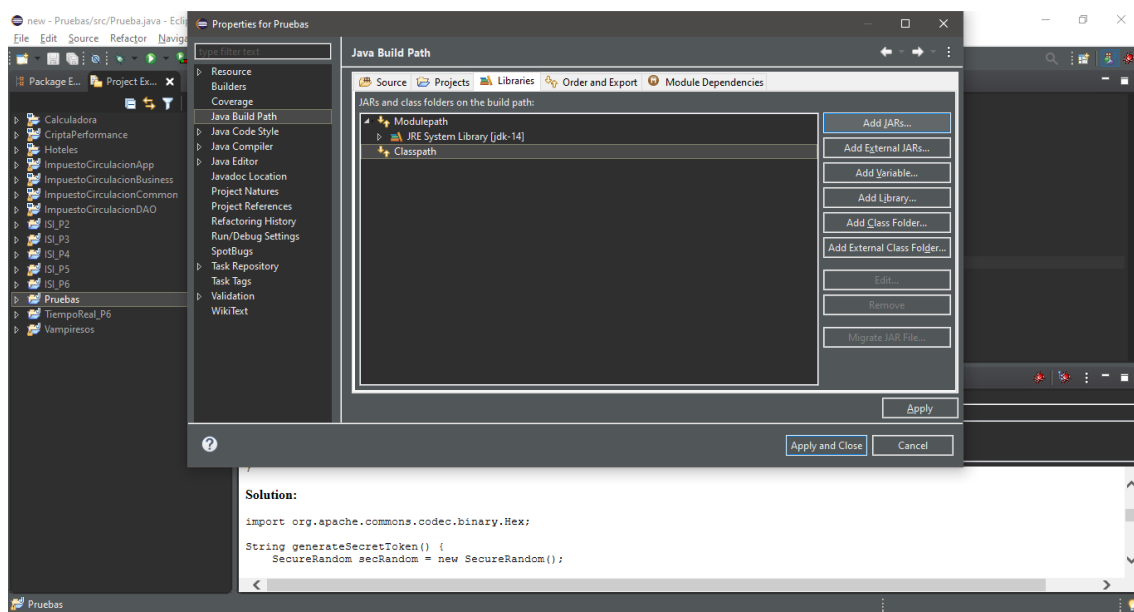
```

<https://repo1.maven.org/maven2/commons-codec/commons-codec/1.10/commons-codec-1.10.jar>

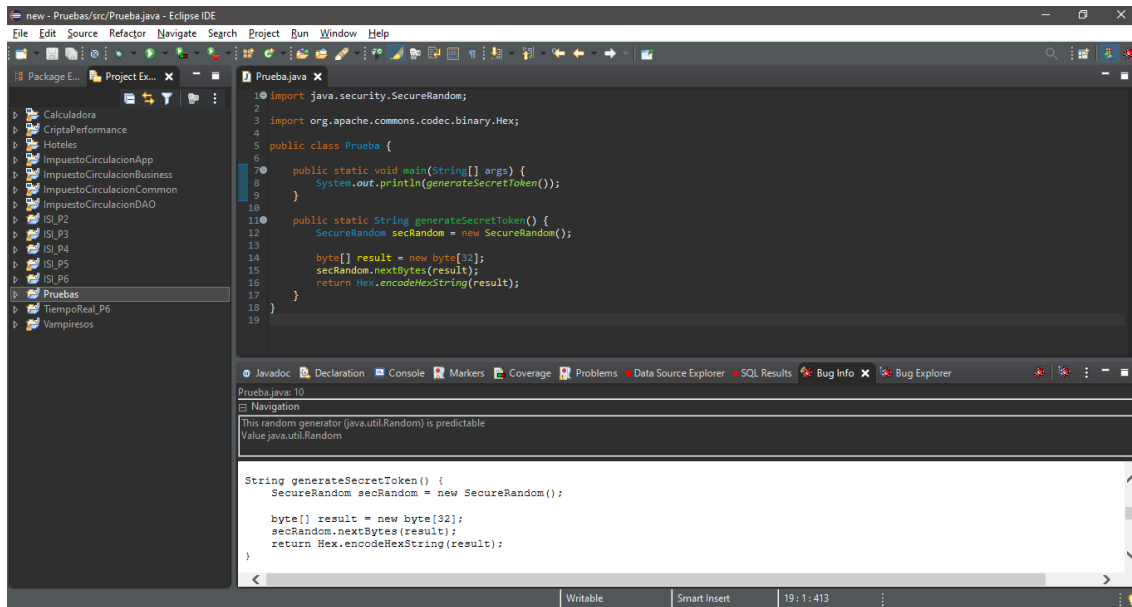
Agregar la librería descargada en el classpath del proyecto. Clic derecho sobre él -> Build Path -> Configure Build Path



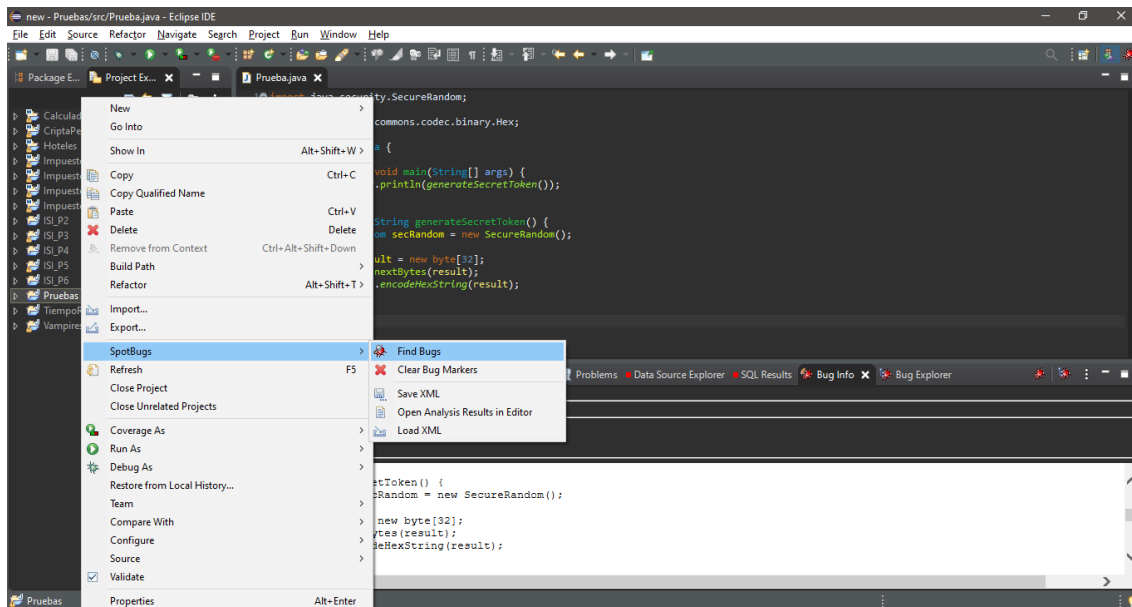
En el Classpath, seleccionar Add JARs y agregarlo.



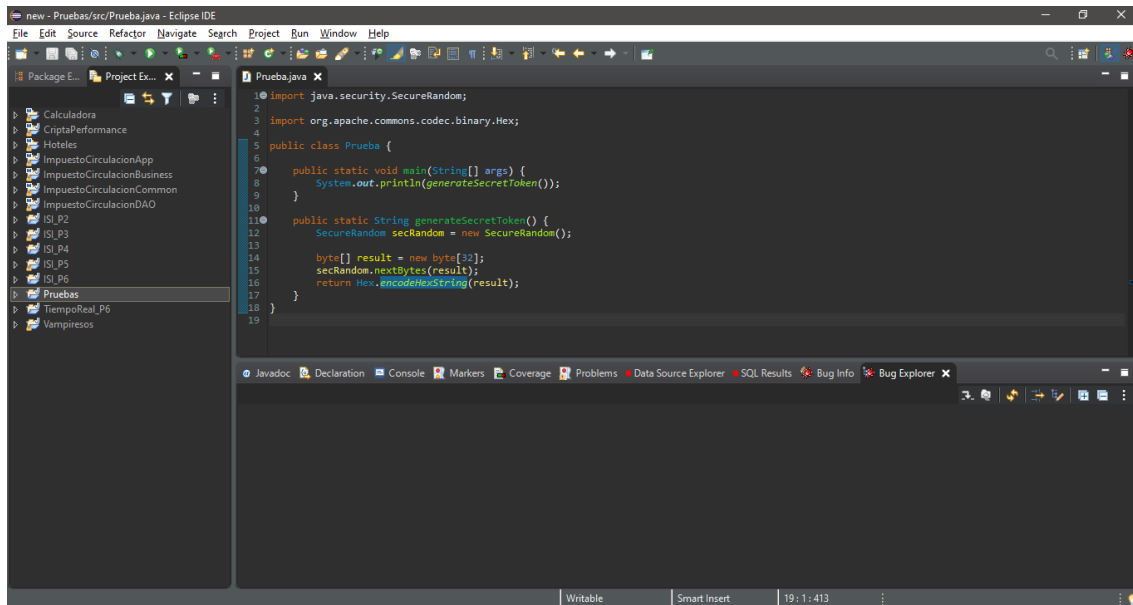
A continuación, se escribe el código seguro o “libre de bugs”.



Dar clic derecho nuevamente sobre el proyecto -> SpotBugs -> Find Bugs

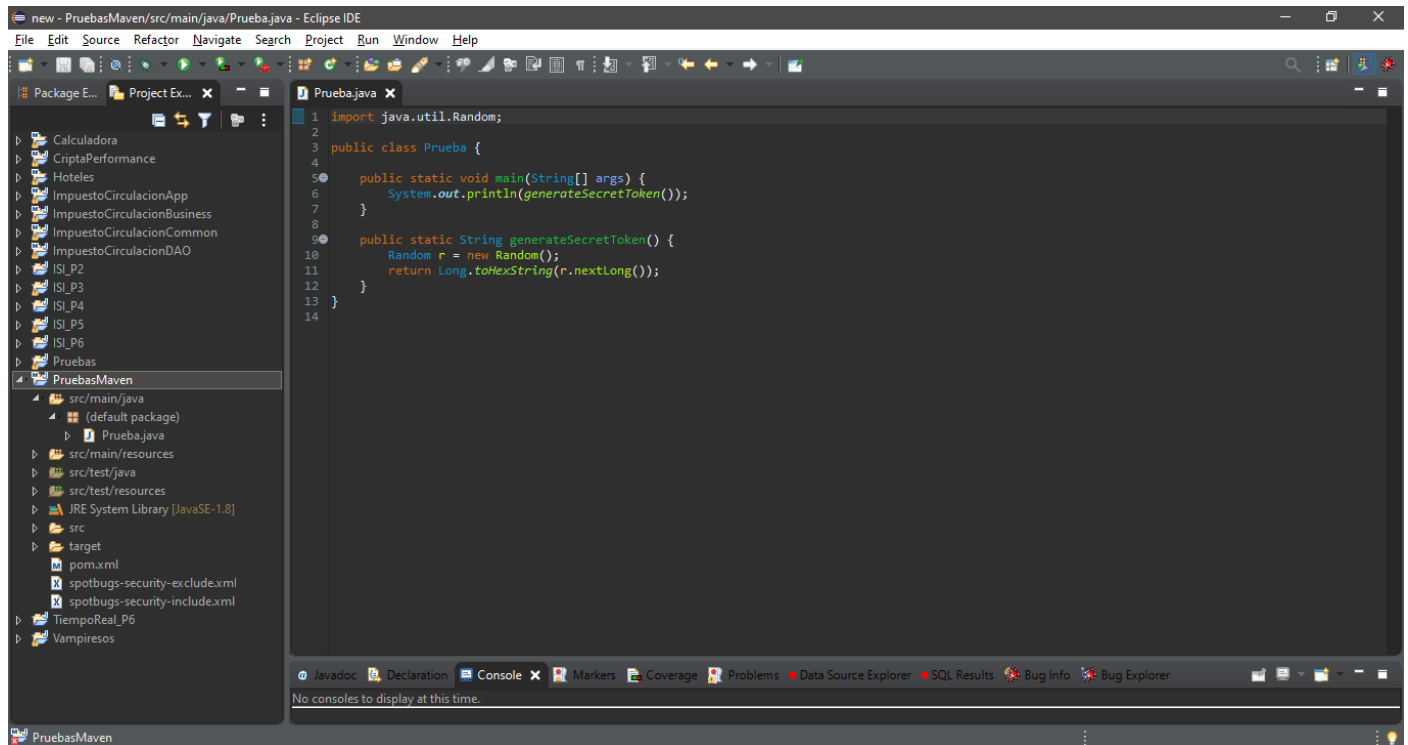


Ahora no se encuentra ningún error de seguridad en el proyecto.

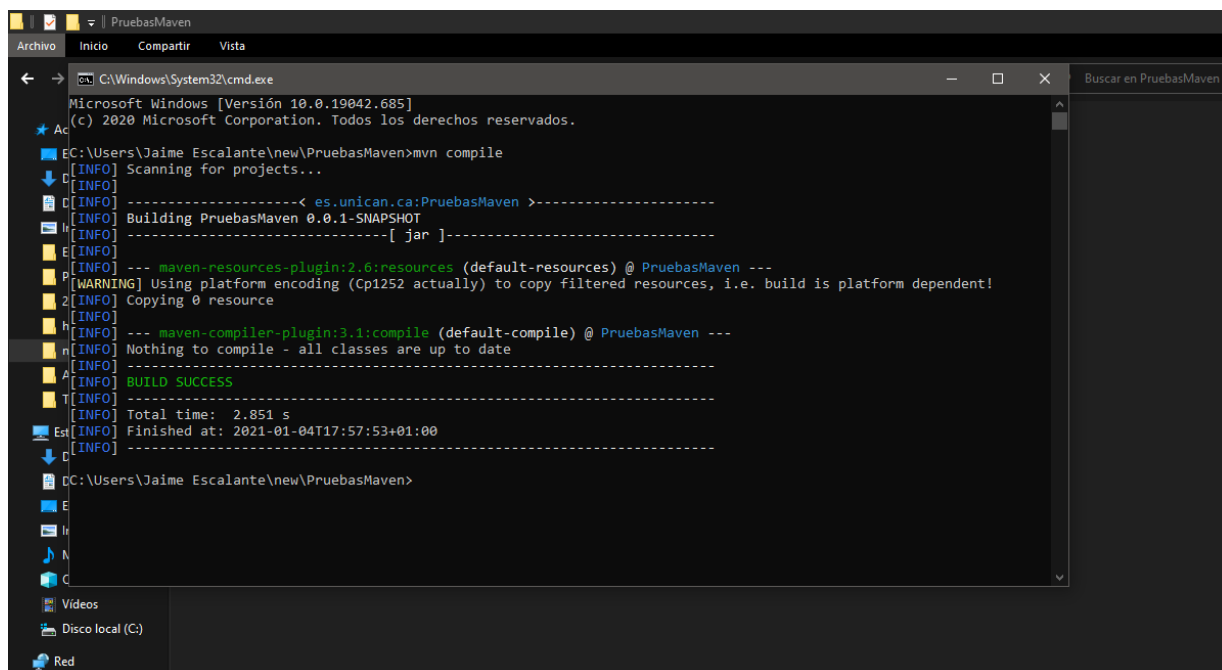


2. Ejemplo guiado con plugin de Maven

Se tiene el mismo ejemplo que en caso anterior, solo que ahora se parte de un proyecto Maven, con el fichero pom.xml definido según el manual de instalación.

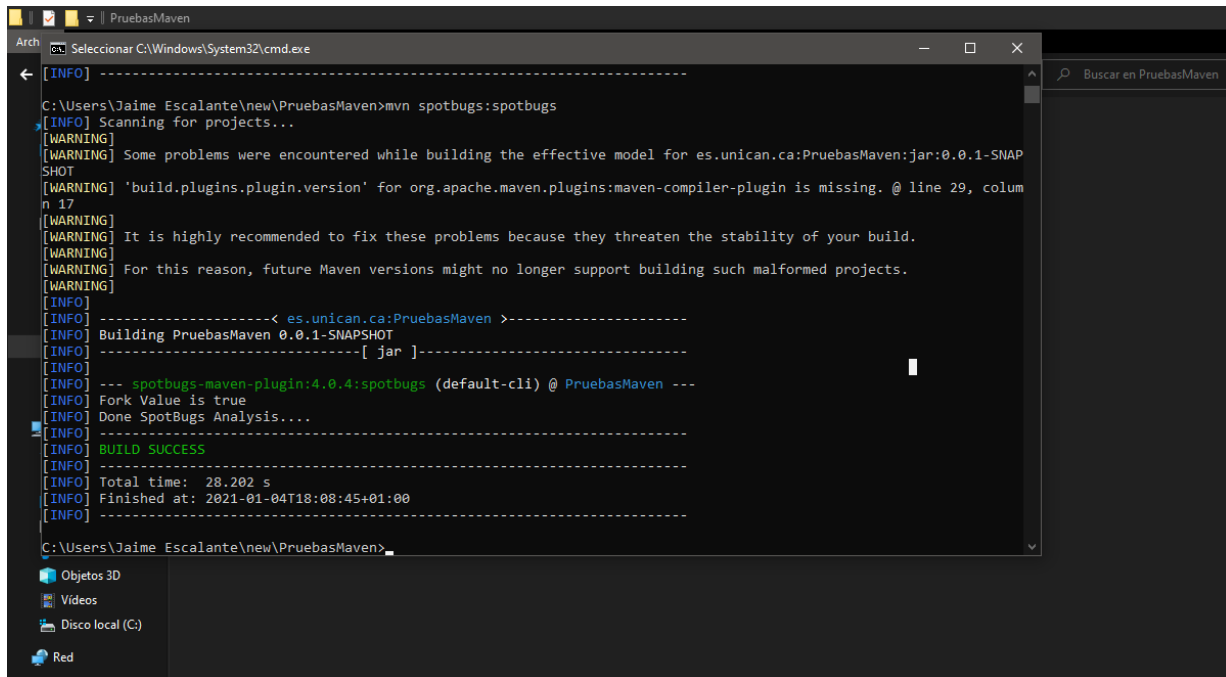


En la ventana de comandos, primero se compila el proyecto, para lo cual se escribe **mvn compile**



A continuación, para ejecutar el análisis escribir **mvn spotbugs:spotbugs**.

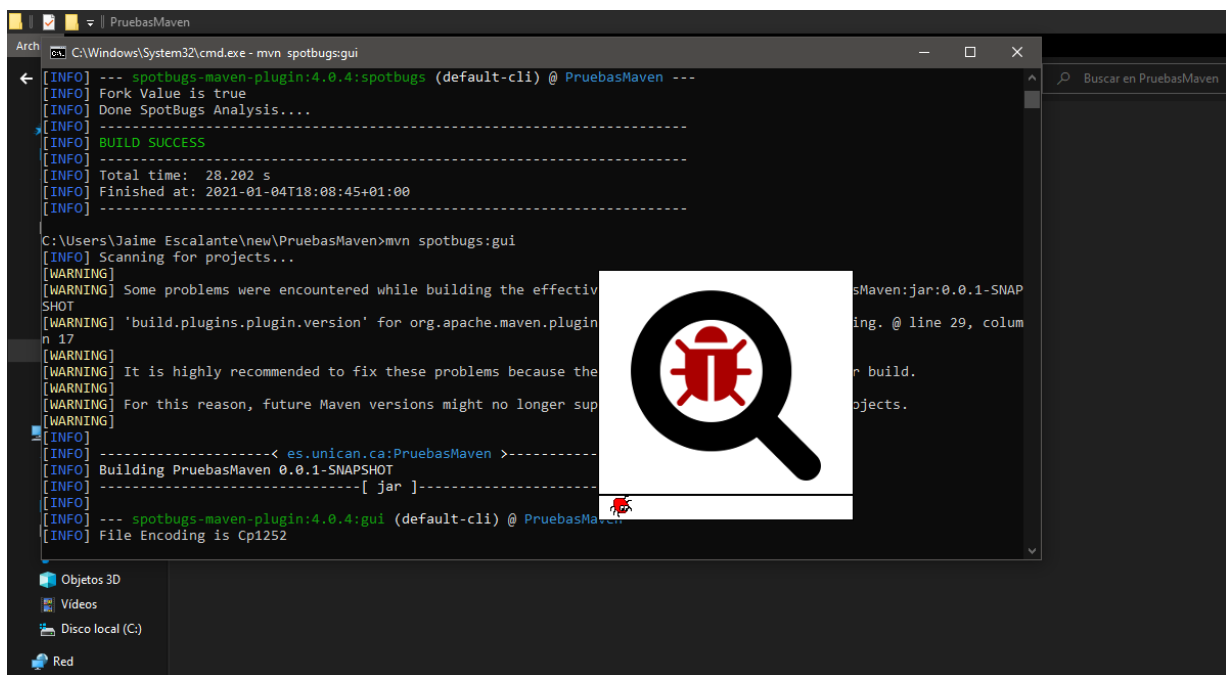
La primera vez que se ejecuta este comando podrán descargarse una serie de librerías.



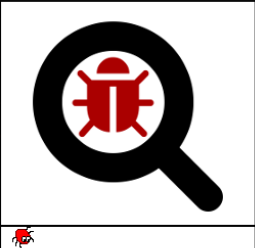
```
Arch C:\Windows\System32\cmd.exe
[INFO] -----
C:\Users\Jaime Escalante\new\PruebasMaven>mvn spotbugs:spotbugs
[INFO] Scanning for projects...
[WARNING] Some problems were encountered while building the effective model for es.unican.ca:PruebasMaven:jar:0.0.1-SNAPSHOT
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-compiler-plugin is missing. @ line 29, column 17
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO] -----< es.unican.ca:PruebasMaven >-----
[INFO] Building PruebasMaven 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- spotbugs-maven-plugin:4.0.4:spotbugs (default-cli) @ PruebasMaven ---
[INFO] Fork Value is true
[INFO] Done SpotBugs Analysis....
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 28.202 s
[INFO] Finished at: 2021-01-04T18:08:45+01:00
[INFO] -----
C:\Users\Jaime Escalante\new\PruebasMaven>
```

Para poder acceder a los resultados mediante una interfaz de usuario, escribir **mvn spotbugs:gui**.

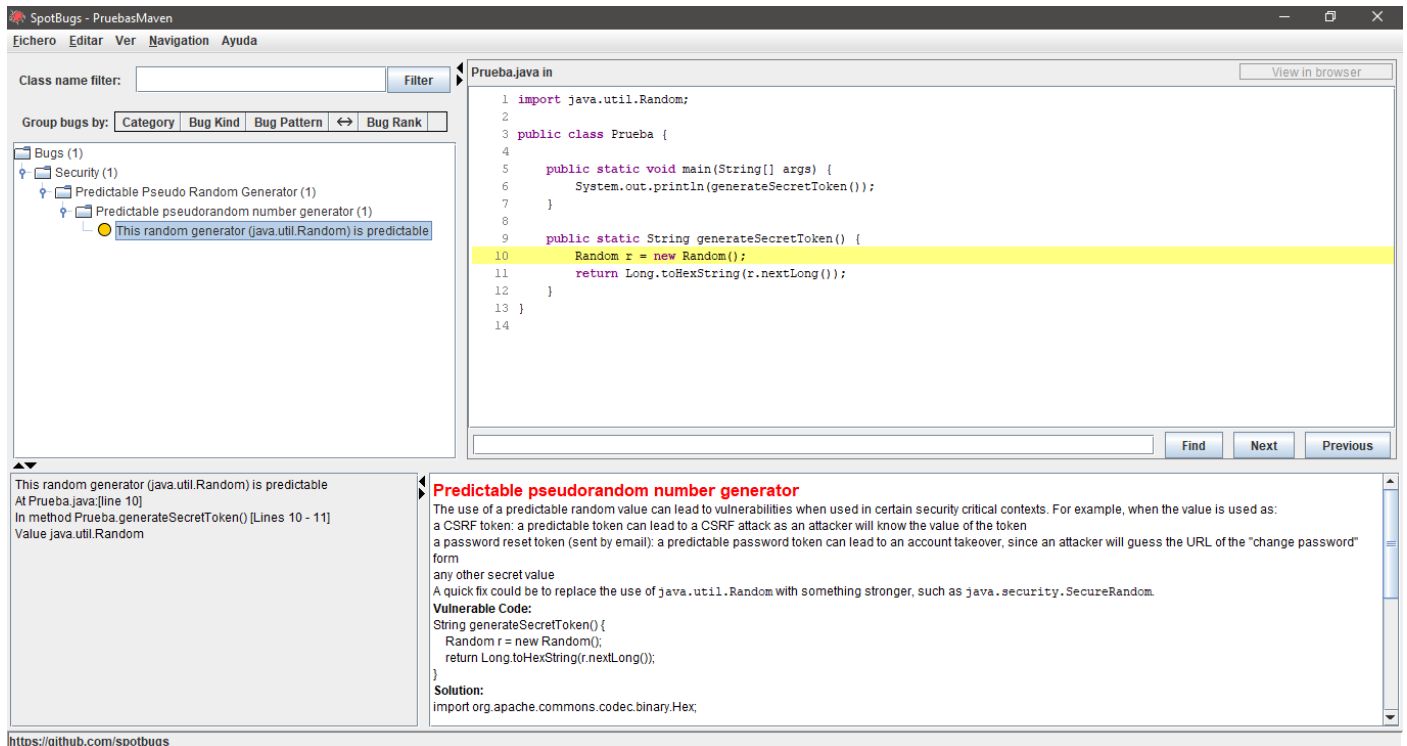
La primera vez que se ejecuta este comando podrán descargarse una serie de librerías.



```
Arch C:\Windows\System32\cmd.exe - mvn spotbugs:gui
[INFO] --- spotbugs-maven-plugin:4.0.4:spotbugs (default-cli) @ PruebasMaven ---
[INFO] Fork Value is true
[INFO] Done SpotBugs Analysis....
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 28.202 s
[INFO] Finished at: 2021-01-04T18:08:45+01:00
[INFO] -----
C:\Users\Jaime Escalante\new\PruebasMaven>mvn spotbugs:gui
[INFO] Scanning for projects...
[WARNING] Some problems were encountered while building the effective model for es.unican.ca:PruebasMaven:jar:0.0.1-SNAPSHOT
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-compiler-plugin is missing. @ line 29, column 17
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO] -----< es.unican.ca:PruebasMaven >-----
[INFO] Building PruebasMaven 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- spotbugs-maven-plugin:4.0.4:gui (default-cli) @ PruebasMaven ---
[INFO] File Encoding is Cp1252
```



Los resultados se muestran en una ventana como la siguiente, con una estructura y contenido similares al caso anterior.



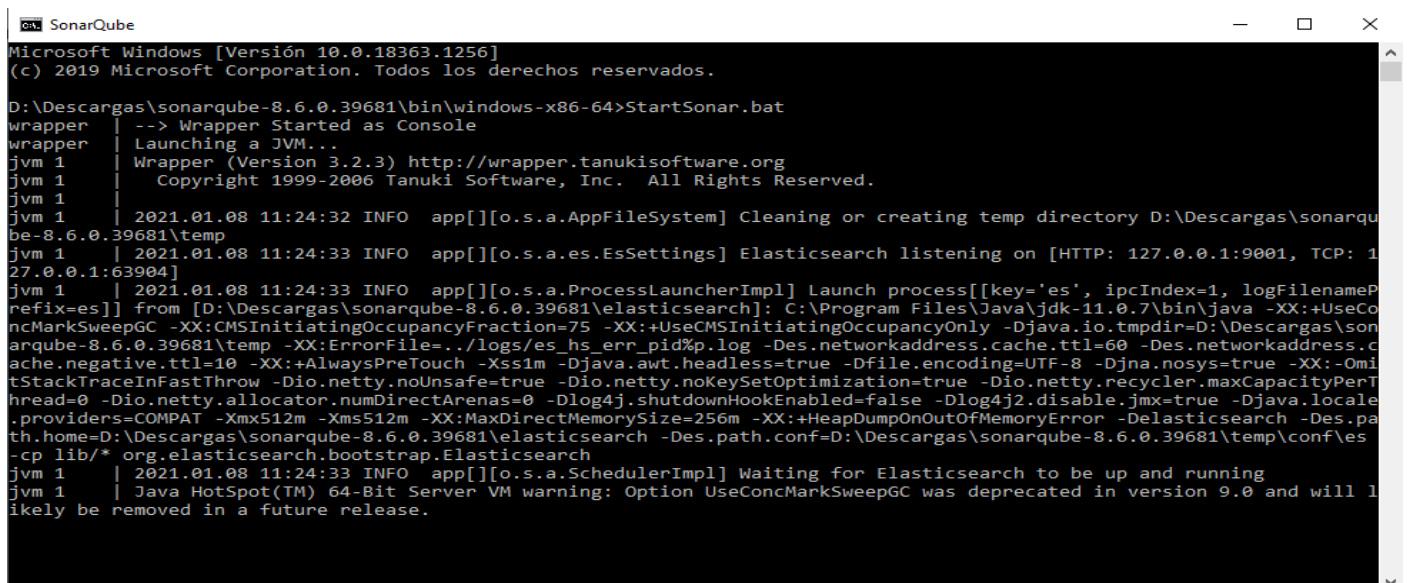
El proceso para solucionar el bug es equivalente al del apartado previo.

3. Ejemplo guiado con SonarQube

En este ejemplo lanzaremos un análisis de Sonar con el plugin FindSecurityBugs sobre el mismo proyecto Maven de los dos ejemplos anteriores. Este ejemplo está enfocado al sistema operativo Windows.

Para poder seguir este ejemplo es necesario del software y la configuración sobre Sonar descrita en el documento [Find Security Bugs - Instalación.pdf](#).

El primer paso será arrancar nuestro servidor local de Sonar, para ello nos dirigiremos al directorio donde tengamos descargado Sonar y colocándonos en la carpeta `/bin/windows-x86-64`, abriremos una terminal desde la que arrancaremos el servidor escribiendo **startsonar.bat**.

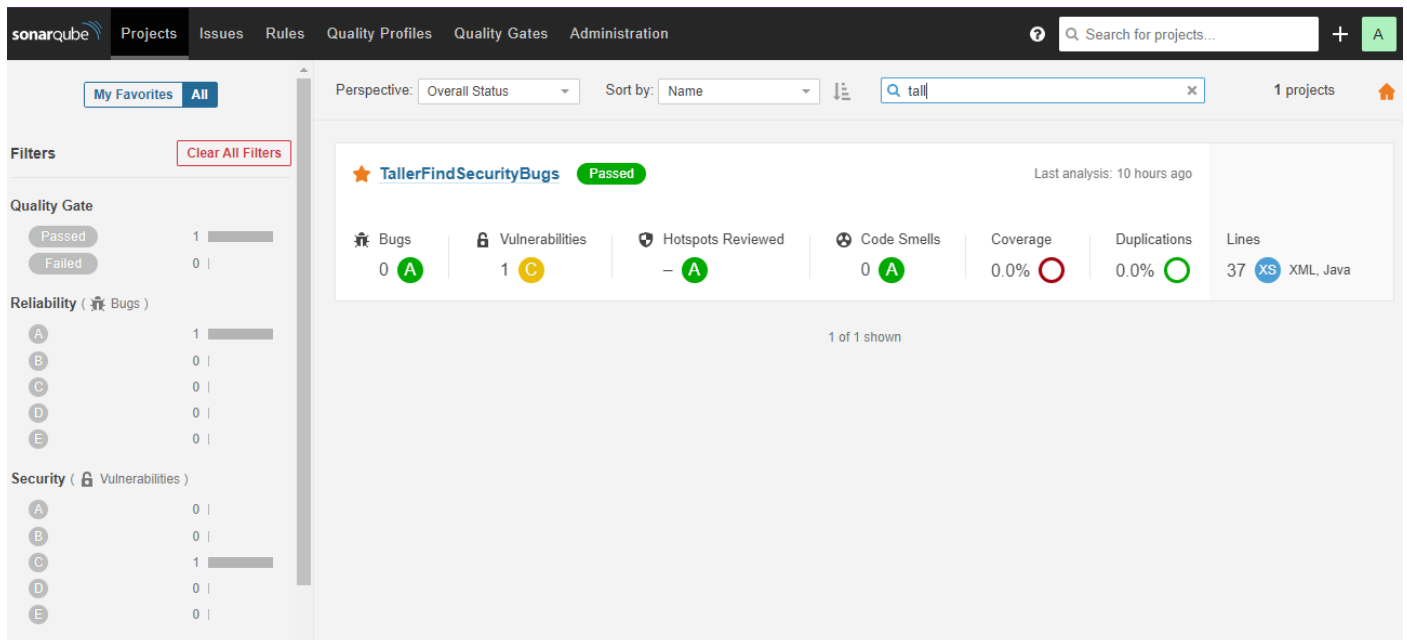


Una vez arrancado el servidor el siguiente paso es lanzar el análisis de Sonar con Maven, para ello nos situaremos en el directorio donde está el proyecto, dentro de la carpeta del proyecto abriremos una terminal, y con el comando **mvn sonar:sonar**, lanzamos el análisis de Sonar. Luego de lanzar el análisis deberíamos obtener el siguiente mensaje.

```
C:\Windows\System32\cmd.exe
[INFO] 1/1 source files have been analyzed
[INFO] Sensor VB.NET Properties [vbnet]
[INFO] Sensor VB.NET Properties [vbnet] (done) | time=2ms
[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=7ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=8ms
[INFO] SCM Publisher SCM provider for this project is: git
[INFO] SCM Publisher 2 source files to be analyzed
[INFO] SCM Publisher 2/2 source files have been analyzed (done) | time=58ms
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=0ms
[INFO] Analysis report generated in 39ms, dir size=77 KB
[INFO] Analysis report compressed in 66ms, zip size=14 KB
[INFO] Analysis report uploaded in 1662ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=es.unican.ca%3APruebasMaven
[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://localhost:9000/api/ce/task?id=AXbhph80yM3vVnLsw7KZ
[INFO] Analysis total time: 11.050 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.651 s
[INFO] Finished at: 2021-01-08T11:58:49+01:00
[INFO] -----
```

Una vez finalizado el análisis, con ayuda de un navegador nos dirigiremos a la dirección <http://localhost:9000/> que la dirección donde por defecto se aloja el servidor de Sonar, nos pedirá un usuario y una contraseña, por defecto las dos son “admin” (es posible que durante el proceso de instalación hayáis tenido que cambiar la contraseña porque lo pide el propio sonar).

Dentro de Sonar, nos dirigiremos a la pestaña proyectos y debería de aparecernos el proyecto que acabamos de analizar.



Como vemos nos indica que el código tiene una vulnerabilidad. Para conocer más detalles acerca de esta vulnerabilidad, podemos pulsar encima del nombre del proyecto y nos redirigirá a una ventana con más información acerca del análisis que ha hecho Sonar. En concreto nos interesa el apartado de “Issues” de esta ventana.

TallerFindSecurityBugs master

Overview Issues Security Hotspots Measures Code Activity

Project Settings Project Information

My Issues All

Bulk Change to select issues to navigate 1 / 1 issues 0 effort

Filters

Type

- Bug 0
- Vulnerability 1
- Code Smell 0

Severity

- Blocker 0
- Critical 0
- Major 1
- Minor 0
- Info 0

src/.../java/es/unican/ca/clasesconbug/ClaseConBug.java

This random generator (java.util.Random) is predictable Why is this an issue? 4 days ago L8

Vulnerability Major Open Not assigned Comment

1 of 1 shown

En este apartado nos indica el tipo de vulnerabilidad que ha detectado en nuestro código, si pinchamos encima de del recuadro rosa, nos redirigirá al código que causa esta vulnerabilidad.

TallerFindSecurityBugs src/main/java/es/unican/ca/clasesconbug/ClaseConBug.java See all issues in this file

```

1  package es.unican.ca.clasesconbug;
2
3  import java.util.Random;
4
5  public class ClaseConBug {
6
7      String generateSecretToken() {
8          Random r = new Random();
9
10         return Long.toHexString(r.nextLong());
11     }
12 }
13

```

This random generator (java.util.Random) is predictable Why is this an issue? 4 days ago L8

Vulnerability Major Open Not assigned Comment

Además, si pulsamos sobre el texto “Why is this an issue?”, aparecerá un recuadro en la parte inferior de la pantalla que nos indicará porque esto supone una vulnerabilidad en nuestro código y la manera en la que podemos solucionarlo.

Security - Predictable pseudorandom number generator

Security - Predictable pseudorandom number generator

Vulnerability Major cwe Available Since Jan 04, 2021 Find Security Bugs (Java)

The use of a predictable random value can lead to vulnerabilities when used in certain security critical contexts. For example, when the value is used as:

- a CSRF token: a predictable token can lead to a CSRF attack as an attacker will know the value of the token
- a password reset token (sent by email): a predictable password token can lead to an account takeover, since an attacker will guess the URL of the “change password” form
- any other secret value

A quick fix could be to replace the use of `java.util.Random` with something stronger, such as `java.security.SecureRandom`.

Vulnerable Code:

```

String generateSecretToken() {
    Random r = new Random();
    return Long.toHexString(r.nextLong());
}

```

Solution:

```

import org.apache.commons.codec.binary.Hex;

String generateSecretToken() {
    SecureRandom secRandom = new SecureRandom();

    byte[] result = new byte[32];
    secRandom.nextBytes(result);
    return Hex.encodeHexString(result);
}

```

Como se puede observar la solución es la misma expuesta en los anteriores ejemplos.