Entornos de Desarrollo: PRO 2 – Sonarlint

Ejercicio 1:

Vulnerabilities:

1- "Credentials should not be hard-coded": Los datos sensibles no deben estar escritos tal cual o incrustados en el código (hard-coded). Lo mejor es codificarlo extrayéndolo de algún fichero externo.

2- "Database queries should not be vulnerable to injection attacks": Las consultas a las bases de datos no deben ser directas y se deben filtrar para evitar "database injections" (o injecciones de bases de datos). Si no, cualquiera podría consultar nuestros datos.

3- "Cipher algorithms should be robust": Los algoritmos de cifrado deben ser robustos para evitar que información sensible se vea comprometida, asegurando así la seguridad y confidencialidad de nuestras comunicaciones.

Bugs:

1- ""wait(...)" should be used instead of "Thread.sleep(...)" when a lock is held": Si un hilo tiene bloqueada la ejecución síncrona mediante un *lock* (ningún otro hilo puede utilizar métodos marcados como *synchronized* y se utiliza el método sleep() en lugar del wait() ningún otro hilo puede adquirir el *lock*, y por tanto la ejecución se queda bloqueada para el resto de hilos. Básicamente, puede dar lugar a interbloqueos o puntos muertos.

2- "Recursion should not be infinite": Aunque el título es bastante descriptivo, la recursividad no debe ser infinita porque esto da lugar a un desbordamiento de la pila (StackOverflowError).

3- "Resources should be closed": Pese a que Java tiene el archiconocido Garbage Collector (recolector de basura), debemos cerrar todos los recursos una vez se hayan utilizado y no se vayan a volver a usar, como por ejemplo los objetos de tipo Scanner (scanner.close()).

Security Hotspots:

1- "Hard-coded secrets are security-sensitive": En la línea del punto 1 de las vulnerabilidades, los secretos codificados directamente en los programas. Por ello, deben estar almacenados fuera del código fuente del programa.

2- "Using non-standard cryptographic algorithms is security-sensitive": Está claro que no usar algoritmos de criptografía estándar puede causar fallas en la seguridad porque los atacantes pueden ser capaces de romper el algoritmo y acceder a los datos.

3- "Using long-term access keys is security-sensitive": Cuanto mayor sea la duración de una clave de acceso, mayor es el tiempo de exposición de dicha clave, y por tanto más tiempo tienen los atacantes para tratar de descifrarla.

Code Smells:

1- "Future keywords should not be used as names": Otra regla bastante descriptiva. La idea es no utilizar futuras palabras marcadas como *keyword* (de cara a futuro) como nombres de nuestras variables. Esto puede causar que nuestro código falle al cambiar de versión de Java.

2- "Conditionals should start on new lines": Con tal de evitar la confusión y los errores, los condicionales deben empezar en nuevas líneas. De otro modo, puede parecer que la sentencia está incompleta y puede causar errores.

3- "Class names should not shadow interfaces or superclasses": Puesto que mientras estén en diferentes paquetes (package) las clases pueden tener el mismo nombre, esto puede dar lugar a algunos problemas cuando una clase y de la que hereda o la que implementa se llaman igual. Puede no quedar claro qué clase está siendo referenciada en el código y da lugar a ambigüedades.

Ejercicio 2:

Recomendaciones de Sonarlint:

1- A "NullPointerException" could be thrown; "o2" is nullable here.
Puesto que el método getObject devuelve un null, en las llamadas tanto al objeto 'o' como 'o2' puede (de hecho, es así) darse una NullPointerException.

Solución: sustituir el "return null" de getObject por "new Object()".

```
private static Object getObject() {
    return new Object();
}
```

2- Replace this use of System.out by a logger.
Sonarlint desaconseja el uso de System.out para imprimir por pantalla y aconseja usar un logger().

Solución: usar dicho logger().

```
Logger logger = Logger.getLogger(Flow.class.getName());
logger.info("This is an o: " + o2.getClass());
```

3- Move this file to a named package.
Se desaconseja el uso del paquete "default" y se aconseja el uso de uno más descriptivo.

Solución: Crear un paquete con el nombre "flow" y mover la clase al paquete.