METODOLOGIAS DE DESARROLLO SEGURO

GRADO EN INGENIERIA DE LA CIBERSEGURIDAD

CURSO ACADÉMICO 2024-2025

CONVOCATORIA 2Q

PRACTICA 1

AUTOR(A): Suárez Suárez, Juan Antonio

Tabla de contenido

1. Buenas prácticas de desarrollo

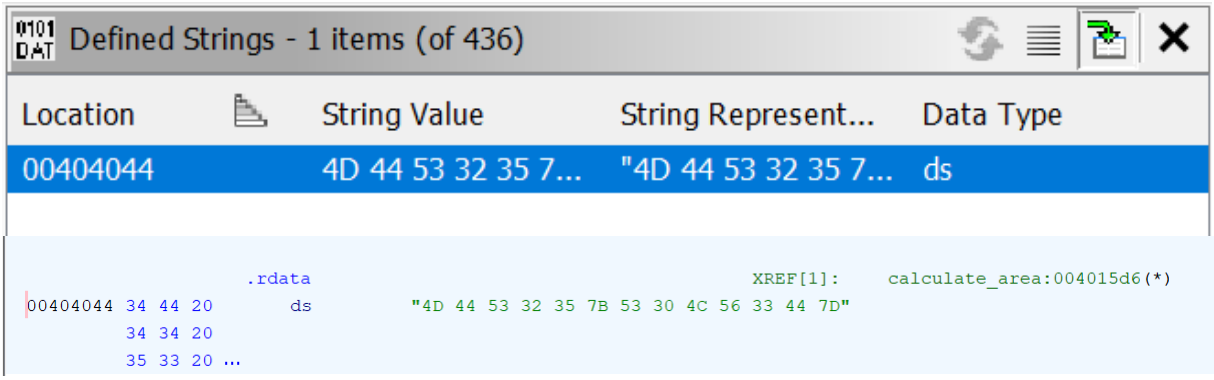
1.1 Ejercicio 1 (CWE-252(Unchecked Return Value))

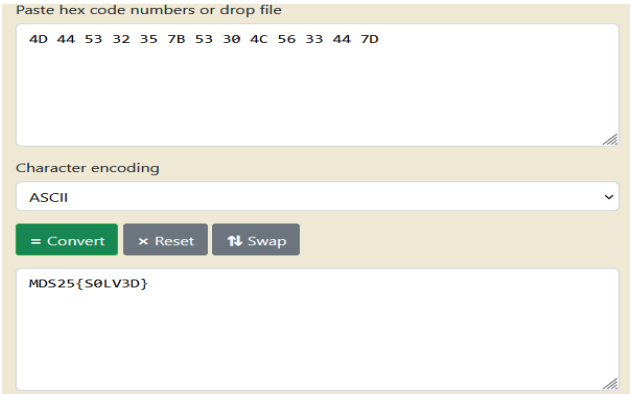
- .cpp: La función scanf se utiliza para leer valores de entrada sin comprobar si la lectura fue exitosa. Si el usuario introduce datos inválidos o se produce un error, se podría continuar utilizando variables sin el valor esperado, lo que puede conducir a comportamientos inesperados o errores posteriores.
- Solución: Verificar el valor devuelto por scanf y en caso de error, gestionar la situación.

  Ejemplo de solución:

```
if (scanf("%d", &ancho) != 1) {
    fprintf(stderr, "Error al leer el ancho.\n");
    return 1;
}

    print(year)
```

- .exe: Para este archivo utilizaremos la herramienta Ghidra para analizar este ejecutable y encontrar el secreto. Podemos observar una cadena en hexadecimal y el formato podría sugerir que sea ASCII.





1.2 Ejercicio 2

- **Línea:** 27 (en el método incrementVisitCount)
  - **CWE:** CWE-362 – *Race Condition*
    - **Explicación:**
      El método incrementVisitCount se ejecuta en múltiples hilos de forma concurrente. La lectura y escritura al archivo visit_count.txt se realizan sin ningún mecanismo de sincronización, lo que puede provocar condiciones de carrera e inconsistencias en el valor del contador (actualizaciones perdidas o sobrescritas).
    - **Solución:**
      Implementar un mecanismo de sincronización para garantizar que la lectura y escritura se realicen de forma sincronizada.
    - Ejemplo de solución:

```
public synchronized static void incrementVisitCount() {
    // Resto.
}
```

1.3 Ejercicio 3

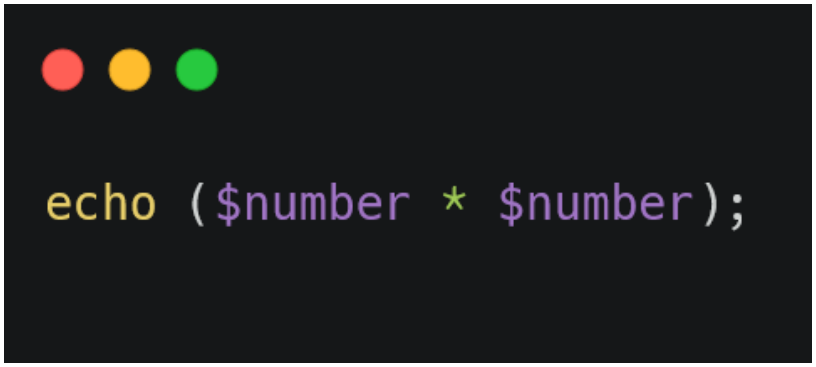
- **Línea:** 20 (dentro de la función create_user)
  - **CWE:** CWE-327 – Use of a Broken or Risky Cryptographic Algorithm
  - **Explicación:**
    Se utiliza el algoritmo MD5 para hashear las contraseñas. MD5 es considerado obsoleto y vulnerable a ataques de colisión, lo que compromete la seguridad de las contraseñas almacenadas.
  - **Solución:**
    Utilizar algoritmos de hash diseñados para el manejo de contraseñas, como bcrypt para fortalecer la seguridad.
  - **Ejemplo de solución (bcrypt):**

```
import bcrypt

def create_user(username, password):
    hashed_password = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()
```

1.4 Ejercicio 4

- Línea: 37
  - CWE: CWE-95 – Improper Neutralization of Directives in a Code Block (Code Injection)

- Explicación:
  Se utiliza la función eval() para ejecutar código construido dinámicamente. Aunque en este caso se evalúa una operación aritmética controlada, el uso de eval() es una mala práctica ya que, si en el futuro se llega a incorporar entrada del usuario, se podría abrir la puerta a ataques de inyección de código.
- Solución:
  Eliminar el uso de eval() y reemplazarlo por una operación directa. Por ejemplo, para mostrar el resultado de la multiplicación, se puede utilizar:



- Principios de diseño saltados:
  - Violación del principio DRY (Don't Repeat Yourself):
    Se repite de forma innecesaria la misma estructura de control (bloques if y bucles for) para cada posible valor, lo que dificulta el mantenimiento y la escalabilidad del código.
  - Falta de modularidad y separación de responsabilidades:
    La lógica de verificación y de impresión se encuentra dispersa en bloques repetitivos en lugar de estar centralizada en funciones o métodos, lo que afecta la claridad y reutilización del código.

2. Expresiones regulares

2.1 Ejercicio 1

- **\b** : Marca el inicio en un límite de palabra.
- **\d{4}**: Coincide con exactamente 4 dígitos, lo que típicamente representa un año.
- **\b**: Termina en un límite de palabra.

Este patrón se utiliza para extraer números que tengan exactamente 4 dígitos, asegurando que se trate de un número.

```python
import sys
import re

line = sys.stdin.readline().strip()

pattern = r"\b\d{4}\b"

years = re.findall(pattern, line)

for year in years:
    print(year)
```

2.2 Ejercicio 2

- **\b**: Inicio en un límite de palabra.
- **(?:E[-]?)?**: Grupo opcional que puede comenzar con la letra "E" seguida opcionalmente de un guion o un espacio. Esto es para casos en los que la placa puede tener un prefijo.
- **\d{4}**: Coincide exactamente con 4 dígitos.
- **[-]?**: Permite un guion o espacio opcional entre los dígitos y las letras.
- **[A-Z]{3}**: Coincide con 3 letras mayúsculas, que completan el formato de la placa.
- **\b**: Finaliza en un límite de palabra.

```python
import re

texto = input().strip()

patron = r'\b(?:E[- ]?)?\d{4}[- ]?[A-Z]{3}\b'

coincidencias = re.findall(patron, texto)

for matricula in coincidencias:
    print(matricula)
```

2.3 Ejercicio 3

- **\b**: Asegura que la coincidencia comience en un límite de palabra.
- **(\d{4})**: Captura el año, que son 4 dígitos.
- -: Un guion literal.
- **(\d{2})**: Captura el mes, con exactamente 2 dígitos.
- -: Otro guion literal.
- **(\d{2})**: Captura el día, también 2 dígitos.
- **\b**: Finaliza en un límite de palabra.

Esta expresión se usa para encontrar fechas en formato YYYY-MM-DD. Luego, mediante la función de reemplazo, se reordena la fecha al formato DD.MM.YYYY.

```python
import sys
import re

line = sys.stdin.readline().rstrip('\n')

pattern = re.compile(r'\b(\d{4})-(\d{2})-(\d{2})\b')

def replace_date(match):
    yyyy = match.group(1)
    mm = match.group(2)
    dd = match.group(3)
    return "{}.{}.{}".format(dd, mm, yyyy)

converted_line = re.sub(pattern, replace_date, line)

print(converted_line)
```

2.4 Ejercicio 4

Para alumnos:

- ([a-z]): Captura la inicial del nombre (una letra minúscula).
- \.: Un punto literal.
- ([a-z]{2,}): Captura el apellido, que debe tener al menos dos letras minúsculas.
- \.: Otro punto literal.
- (\d{4}): Captura el año (4 dígitos) de matriculación.
- @alumnos\.urjc\.es: Coincide con la parte fija del email para alumnos.

Para profesores:

- ([a-z]+): Captura el nombre (una o más letras minúsculas).
- \.: Punto literal.
- ([a-z]+): Captura el apellido (una o más letras minúsculas).
- @urjc\.es: Coincide con la parte fija del email para profesores.

La barra vertical | indica que se puede hacer coincidir con uno u otro patrón, permitiendo distinguir entre ambos tipos de emails.

```python
import sys
import re

line = sys.stdin.readline().rstrip('\n')

pattern = re.compile(
    r'([a-z])\.([a-z]{2,})\.(\d{4})@alumnos\.urjc\.es'  # alumno
    r'|'
    r'([a-z]+)\.([a-z]+)@urjc\.es'                      # profesor
)

matches = pattern.finditer(line)

for match in matches:
    if match.group(1) is not None:
        apellido = match.group(2)
        año = match.group(3)
        print(f"alumno {apellido} matriculado en {año}")
    else:
        nombre = match.group(4)
        apellido = match.group(5)
        print(f"profesor {nombre} apellido {apellido}")
```

2.5 Ejercicio 5

- **\b**: Asegura que la coincidencia empiece en un límite de palabra.
- **(?:C/|Calle)**: Grupo sin captura que acepta “C/” o “Calle” al inicio.
- **\s+**: Uno o más espacios.
- **([A-ZÁÉÍÓÚÜÑ][a-záéíóúüñ]+)**: Captura el nombre de la calle: empieza con mayúscula (incluyendo acentuadas) seguida de minúsculas.
- **,?**: Permite una coma opcional después del nombre.
- **\s***: Espacios opcionales.
- **(?:[Nn](?:[º°])?\s*)?**: Grupo opcional para capturar la letra “N” (o “n”) y opcionalmente el símbolo º o °, junto con espacios, que representa "número".
- **(\d+)**: Captura el número de la calle (uno o más dígitos).
- **,**: Una coma literal que separa el número del código postal.
- **\s***: Espacios opcionales.
- **(\d{5})**: Captura el código postal, que debe tener exactamente 5 dígitos.
- **\b**: Termina en un límite de palabra.

```python
import re

regex = r"\b(?:C/|Calle)\s+([A-ZÁÉÍÓÚÜÑ][a-záéíóúüñ]+),?\s*(?:[Nn](?:[º°])?\s*)?(\d+),\s*(\d{5})\b"

texto = input()

coincidencias = re.findall(regex, texto)

for calle, numero, cp in coincidencias:
    print(f"{cp}-{calle}-{numero}")
```

2.6 Ejercicio 6

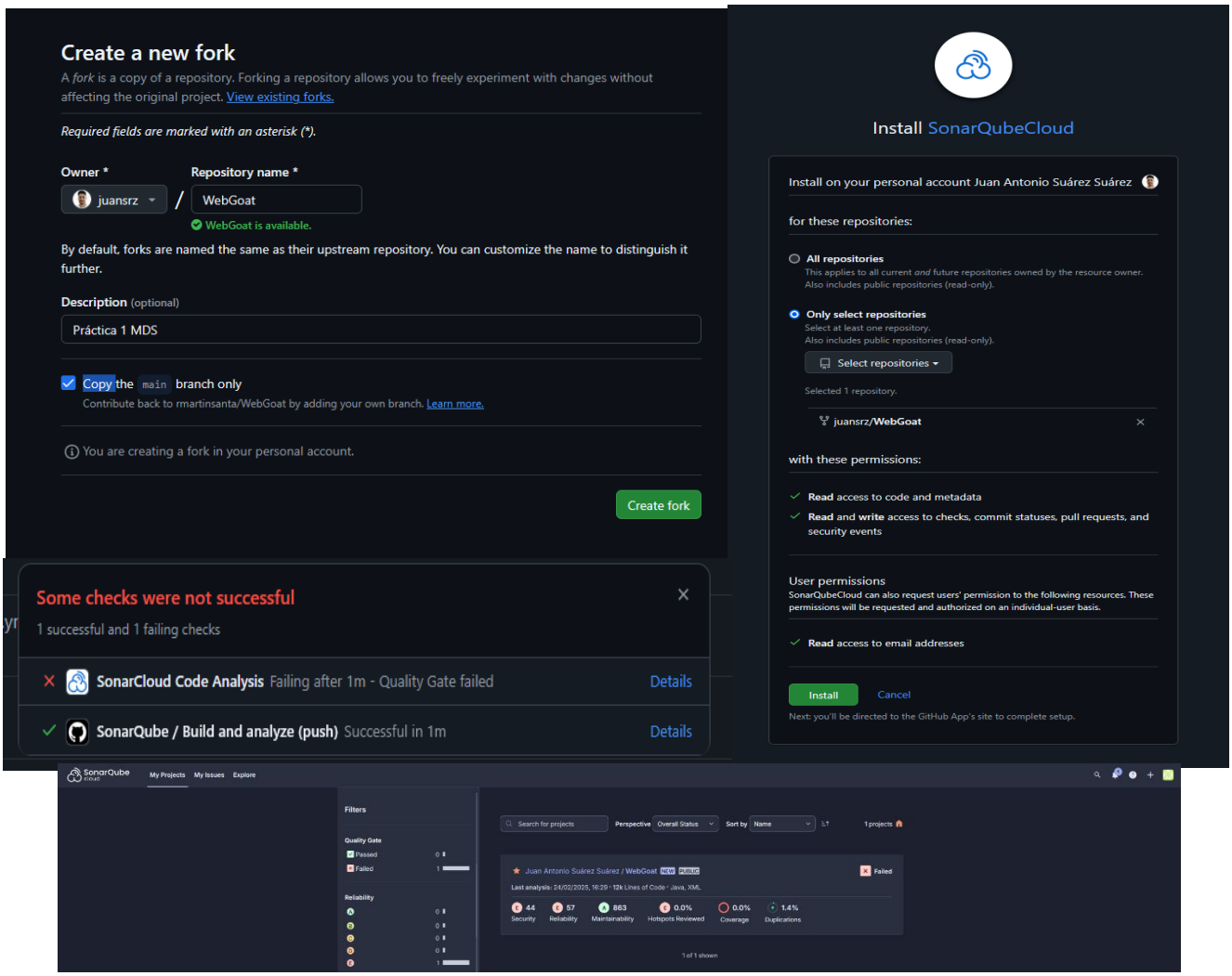
- **^.*?\s+**: Comienza al inicio de la línea y toma lo mínimo necesario (cualquier carácter) hasta encontrar uno o más espacios. Se usa para saltar parte del contenido inicial (por ejemplo, fecha o timestamp).
- **(?P<nivel>\S+)**: Captura en el grupo llamado "nivel" una secuencia de caracteres sin espacios, que normalmente indica el nivel del log (como INFO, ERROR, etc.).
- **\s+\S+\s+---\s+\[**: Después de "nivel", espera espacios, una palabra (posiblemente la fecha/hora), más espacios, y luego la secuencia literal --- [que indica el inicio de la información del hilo.
- **(?P<hilo>[^\]]+)**: Captura en el grupo "hilo" todos los caracteres hasta encontrar el], es decir, el identificador del hilo de ejecución.
- **\]\s+**: Cierra el grupo del hilo con un] y espera espacios.
- **(?P<clase>[^:]+)**: Captura en el grupo "clase" una secuencia de caracteres que no sean dos puntos (:), generalmente el nombre de la clase o módulo.
- **\s*:\s***: Permite espacios opcionales antes y después del :, que separa el nombre de la clase del mensaje.
- **(?P<mensaje>.*)**: Captura en el grupo "mensaje" todo lo que queda en la línea, es decir, el mensaje completo del log.
- **$**: Marca el final de la línea.

```python
import sys
import re

patron = re.compile(r'''
    ^.*?\s+
    (?P<nivel>\S+)
    \s+\S+\s+---\s+\[
    (?P<hilo>[^\]]+)
    \]\s+
    (?P<clase>[^:]+)
    \s*:\s*
    (?P<mensaje>.*)
    $
''', re.VERBOSE)

for linea in sys.stdin:
    linea = linea.rstrip('\n')
    m = patron.match(linea)
    if m:
        nivel = m.group('nivel')
        hilo = m.group('hilo')
        clase = m.group('clase').strip().rsplit('.', 1)[-1]
        mensaje = m.group('mensaje').strip()
        print("\"{}\",\"{}\",\"{}\",\"{}\"".format(nivel, hilo, clase, mensaje))
```

3. Integración continua

En esta parte, se ha implantado un flujo de Integración Continua (CI) en un repositorio de GitHub, usando SonarCloud para el análisis de la calidad del código y la detección de vulnerabilidades.

- https://github.com/juansrz/WebGoat

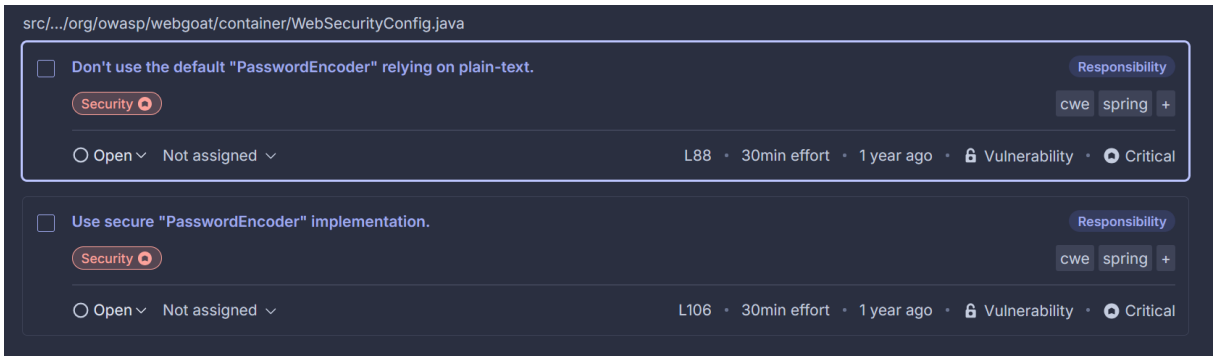


4. Parte 2: Mitigación

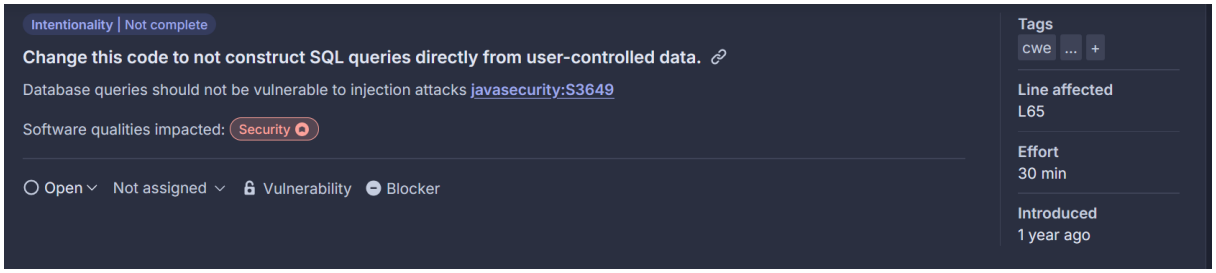
4.1 Elige dos vulnerabilidades (una cada miembro del equipo) de tipo *Blocker* o *Critical*, explica cuáles la vulnerabilidad detectada, por qué ha sido detectada (y si realmente es una vulnerabilidad y no un falso positivo).

- Uso del PasswordEncoder por defecto:
  - Vulnerabilidad: Don't use the default 'PasswordEncoder' relying on plain-text/Use secure "PasswordEncoder" relying on plain-text.
    - Se debe configurar explícitamente un codificador seguro, como BCryptPasswordEncoder, si no puede almacenar o comparar las

contraseñas en texto plano con la configuración por defecto. Podría permitir a un atacante exponer contraseñar o recuperar credenciales.



- SQL injection:
  - Vulnerabilidad: “Change this code to not construct SQL queries directly from user-controlled data.”
    - Se construyen consultas SQL concatenando valores que provienen de la entrada del usuario, esto puede permitir a un atacante introducir sentencias y obtener acceso no autorizado.



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

- Sustitución de PasswordEncoder por BcryptPasswordEncoder:
  - El archivo modificado ha sido: WebSecurityConfig.java
    - Al utilizar BCrypt las contraseñas se hashean de un algoritmo robusto, por lo tanto, aunque se vean comprometidas no se podrán obtener en texto plano. Aumenta la dificultad de ejecución de ataques por fuerza bruta o ataques de diccionario.

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

  - Y también en la configuración global:

```
auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
```

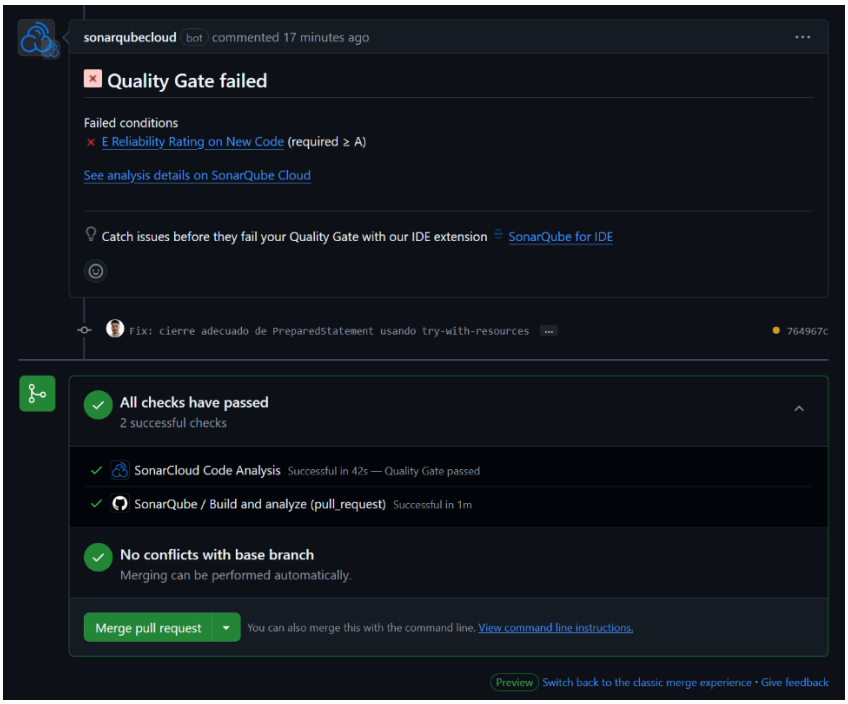
- Uso de Prepared Statements en lugar de concatenaciones:
  - Assignment5.java
    - Usando consultas parametrizadas. Con este método se garantiza que los valores proporcionados por el usuario se traten como datos y no como parte de la sentencia SQL, así se evita la posibilidad de que un atacante inserte código a través de los parámetros.

```
try (var connection = dataSource.getConnection()) {
      PreparedStatement statement = connection.prepareStatement(

              "SELECT password FROM challenge_users WHERE userid = ? AND password = ?"

      );

      statement.setString(1, username_login);

      statement.setString(2, password_login);

      ResultSet resultSet = statement.executeQuery();
```

4.3 Crea una pull Request de la nueva rama a la rama principal. ¿Qué opina Sonar de los cambios realizados?

Al crear la Pull Request desde la rama Fix security vulnerabilities hacia la rama principal, Sonar inicialmente marcó un fallo en el Quality Gate por un problema de “try-with-resources”. Es decir, detectó que en el método donde se usa el PreparedStatement no se cerraban los recursos correctamente.

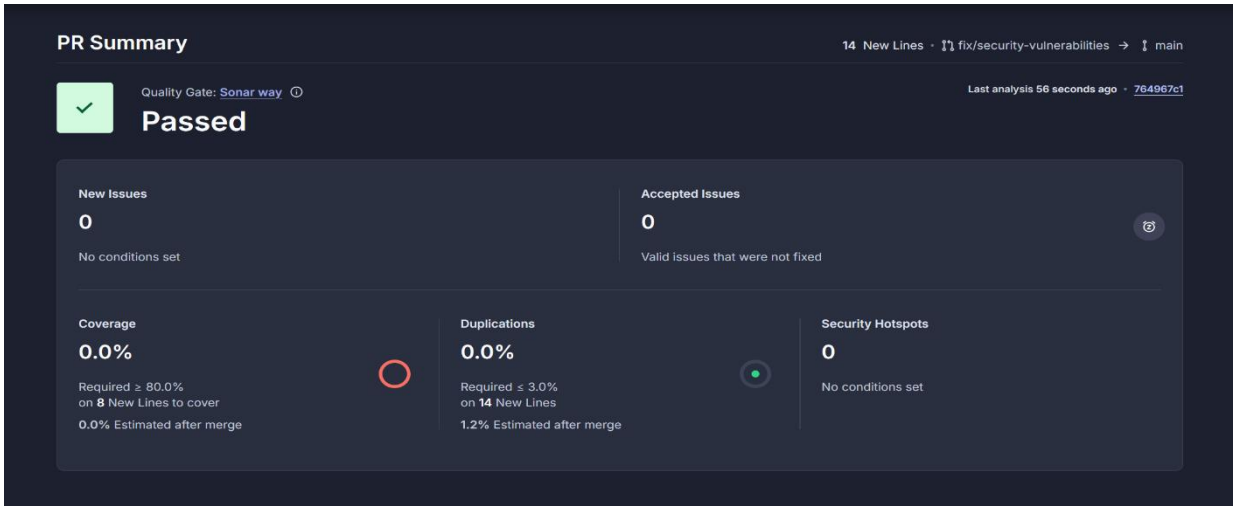
Tras aplicar la sugerencia de Sonar, se volvió a subir un nuevo commit con esa corrección. Al reanalizar el proyecto, Sonar validó los cambios, pasando así el Quality Gate sin incidencias adicionales. Con ello, se solucionó las vulnerabilidades.

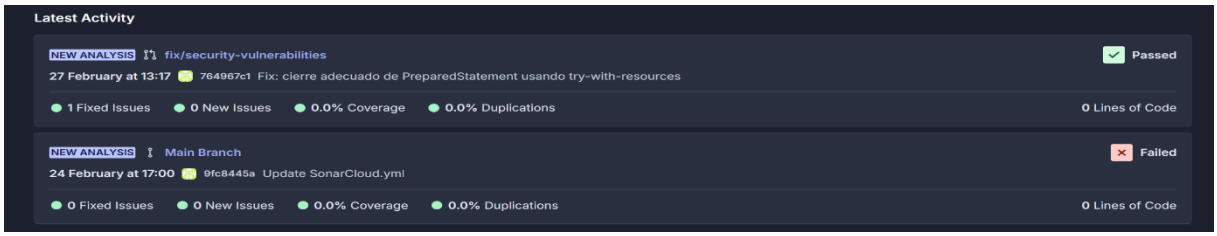


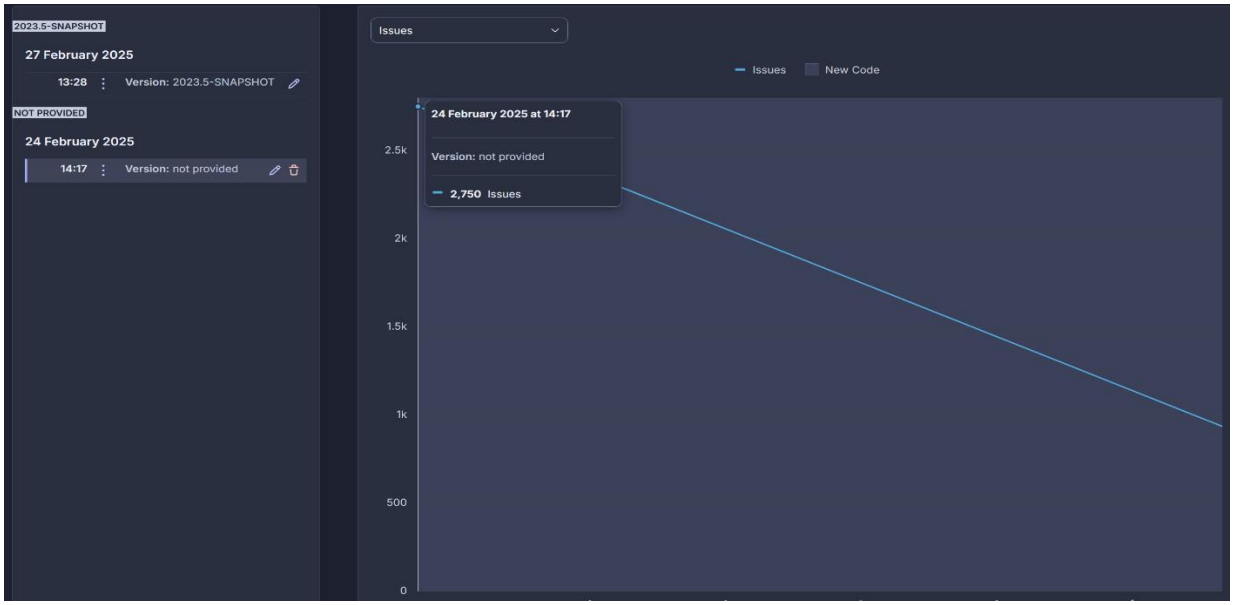
4.4 Junta(merge) la nueva rama con la rama principal. Una vez finalizado el análisis, ¿qué cambios se han producido en el proyecto? ¿Cuántas vulnerabilidades detecta ahora?

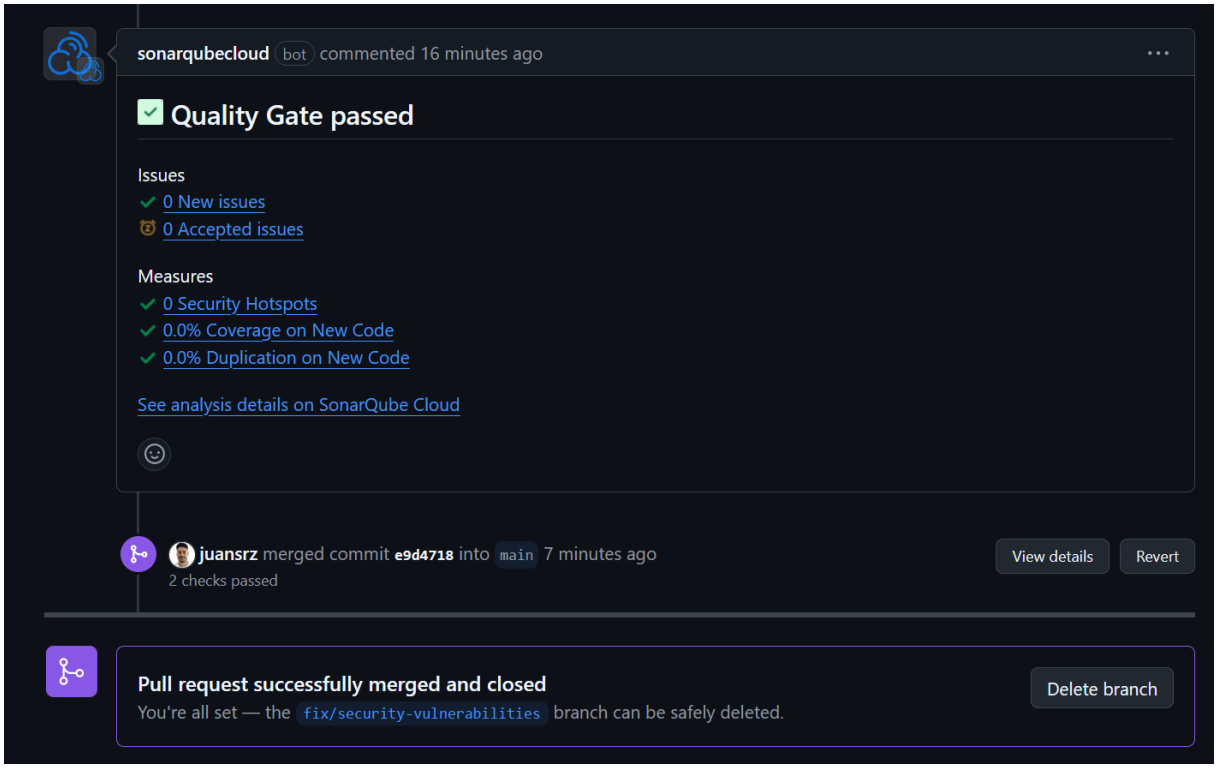
Tras hacer el merge de la rama fix/security-vulnerabilities con la rama principal, el análisis de Sonar muestra ahora lo siguiente:

- El Quality Gate ha pasado correctamente.
- No se detectan nuevas vulnerabilidades en el código modificado y el recuento total de las vulnerabilidades ha bajado en el análisis reciente.
- Se han corregido las vulnerabilidades relacionadas con el uso de un PasswordEncoder inseguro y la inyección de SQL.
- los cambios introducidos en la nueva rama han solventado los fallos de seguridad elegidos y Sonar confirma que el commit cumple con los criterios de calidad y seguridad establecidos para pasar el análisis.




Latest Activity
NEW ANALYSIS fix/security-vulnerabilities
Passed
27 February at 13:17 764967c1 Fix: cierre adecuado de PreparedStatement usando try-with-resources
1 Fixed Issues
0 New Issues
0.0% Coverage
0.0% Duplications
0 Lines of Code
NEW ANALYSIS Main Branch
Failed
24 February at 17:00 9fc8445a Update SonarCloud.yml
0 Fixed Issues
0 New Issues
0.0% Coverage
0.0% Duplications
0 Lines of Code


2023.5-SNAPSHOT
27 February 2025
13:28 Version: 2023.5-SNAPSHOT
NOT PROVIDED
24 February 2025
14:17 Version: not provided
Issues
Issues
New Code
24 February 2025 at 14:17
Version: not provided
2,750 Issues
2.5k
2k
1.5k
1k
500
0


sonarqubecloud bot commented 16 minutes ago
Quality Gate passed
Issues
0 New issues
0 Accepted issues
Measures
0 Security Hotspots
0.0% Coverage on New Code
0.0% Duplication on New Code
See analysis details on SonarQube Cloud
juansrz merged commit e9d4718 into main 7 minutes ago
2 checks passed
View details
Revert
Pull request successfully merged and closed
You're all set — the fix/security-vulnerabilities branch can be safely deleted.
Delete branch