



CURSO 22/23

# PRACTICA 2

## METODOLOGIAS DE DESARROLLO SEGURO GRUPO - U

JOSÉ LUIS MEZQUITA JIMÉNEZ Y MIGUEL ANGEL VILLANUEVA VILLASEÑOR  
INGENIERIA DE LA CIBERSEGURIDAD



## PARTE 1: Análisis estático

1. Dado un texto de una sola línea, determinar todos los años (números formados estrictamente por 4 dígitos) que aparecen. Un año es una cadena numérica de longitud 4, con cualquier valor en el rango [0000, 9999]. Se tendrá que imprimir por pantalla en el lenguaje deseado usando una expresión regular todos los años que vienen en el texto en orden de aparición, en una línea cada uno.

```
4 text= input("")
5 pattern = r"\b\d{4}\b"
6 results = re.findall(pattern, text)
7 for match in results:
8     print(match)
9
```

En esta expresión regular hemos usado para empezar el `\b` para que se distinga un cambio en el tipo de dato, es decir, que aparezca de forma independiente el número de 4 cifras (1234 y no a1234), a continuación, usamos `\d{4}` para especificar que son 4 números y, por último, volvemos a usar `\b` para indicar que queremos que haya un cambio, es decir, que sean independientes y no estén seguidos por más caracteres.

2. Dado un texto de una línea, determinar todas las matrículas que aparecen. Una matrícula es una cadena que tiene las siguientes características: Se tendrá que imprimir por pantalla usando el lenguaje elegido usando una expresión regular todas las matrículas que vienen en el texto en orden de aparición.

```
4 text= input("")
5 pattern = r"\b((E( |-|))?\d{4}(|-|)[A-Z]{3})\b"
6 results = re.findall(pattern, text)
7 for match in results:
8     print(match[0])
```

En esta expresión regular, hemos vuelto a usar el `\b` tanto al principio como al final para “cortar” a la hora de buscar, es decir, que lo que buscamos, en este caso, sea una matrícula independiente (**E1337ZZZ** y no **holaE1337ZZZadidos**). `(E( |-|))?` lo hemos usado para en el principio de la expresión, detectar aquellas matrículas que empiecen por E y continúen con un espacio o un guion o sin espacio, siguiendo después del elemento `?` que nos va a indicar que puede aparecer 1 o 0 veces. `\d{4}` lo hemos usado para indicar que vienen 4 números después. Por último, hemos usado `(|-|)[A-Z]{3}` para indicar que después viene un guion (-) y 3 letras mayúsculas.

3. Dado un formato de fechas yyyy-mm-dd, se pide convertir a dd.mm.yyyy. Para cada match encontrado en los documentos propuestos se tendrá que imprimir en el siguiente formato (los rangos de fecha pueden ser erróneos, pueden existir un mes 20).

```
4 text=input("")
5 pattern = r"\b(\d{4}-\d{2}-\d{2})\b"
6 results = re.findall(pattern, text)
7 for match in results:
8
9     anio, mes, dia = match.split("-")
10    fechaBuena = dia+"."+mes+"."+anio
11    fraseBuena = re.sub(match, fechaBuena, text)
12    text = fraseBuena
13 print(text)
```

En esta expresión regular hemos vuelto a usar `\b` por el mismo motivo explicado anteriormente. Después hemos usado `(\d{4}-\d{2}-\d{2})` para indicar que estamos buscando 4 cifras numéricas (`\d` indica que es un numero), después un guion(-), después dos cifras numéricas, después un guion (-) y por último otras 2 cifras numéricas

Para cada match encontrado. Hemos troceado el match usando la función Split (trocea un string por el carácter que le indiques, en nuestro caso es un guion, y lo almacenamos en año, mes y día, que son tres variables). Después en fechaBuena, concatenamos la fecha uniéndolas con un punto. La función `re.sub` nos va a meter fechaBuena donde se hizo el match en el texto introducido. Por último, imprimimos por pantalla

4. Dado un texto, determinar cuándo se ha encontrado un email de alumno de nuestra universidad "@alumnos.urjc.es" o profesor "@urjc.es". Los emails de alumnos están formados del siguiente patrón (puedes asumir que el input siempre estará en minúscula): Inicial del usuario, seguido de punto. Apellido del usuario, siempre mayor o igual a 2 caracteres. Seguidos de un punto y la fecha de matriculación. Todos finalizan con "@alumnos.urjc.es".

```
4 text= input("")
5 pattern = r"\b([a-z]+\.[a-z]+)(@urjc\.es)|[a-z]{1,2}\.(\d{4})(@alumnos\.urjc\.es)\b"
6 results = re.findall(pattern, text)
7 for match in results:
8     #print(match)
9     if match[0] == '@alumnos.urjc.es':
10        apellido = match[4]
11        anio = match[5]
12        print('alumno '+apellido+' matriculado en '+anio)
13    else:
14        nombre = match[1]
15        apellido = match[2]
16        print('profesor '+nombre+' apellido '+apellido)
17
```

En esta expresión regular hemos vuelto a usar `\b` por el mismo motivo explicado anteriormente. Hemos usado el indicador `|` para decir que haga match con alguno de los dos casos. El primer caso, `([a-z]+\.[a-z]+)(@urjc\.es)` sirve para encontrar los correos de los profesores, donde `[a-z]+` indica una serie de letras minúsculas de cualquier tamaño para indicar el nombre del profesor, y lo mismo con el apellido, y añadimos `@urjc\.es` para determinar que el correo es de un profesor.

El segundo caso, `[a-z]{1,2}\.([a-z]+\)\.(\d{4})(@alumnos\.urjc\.es)` es para encontrar los correos de los alumnos, hemos usado `[a-z]{1,2}` para detectar la inicial o las dos iniciales del nombre del alumno, para el apellido hemos usado `([a-z]+)` igual que antes, para indicar el año hemos usado `\d{4}` que detecta 4 dígitos, y por ultimo para determinar si el correo es de un alumno hemos usado `@alumnos\.urjc\.es`. Luego hemos usado un if comparando el ultimo grupo de la expresión para ver que tipo de correo era.

- Una dirección estará compuesta de una calle representada por "C/" o "Calle" seguido de un espacio con el nombre de la calle (una sola palabra) donde la primera letra debe estar en mayúscula, opcionalmente una coma, un numero arbitrario de espacios, el número en cualquiera de los siguientes formatos (Nº7, nº7, N 7, n 7, 7, Nº 7, nº 7, n7, N7). No es válido N º7, n º7, ni º7, la N podría estar en mayúsculas o minúsculas. Seguido, una coma, un número arbitrario de espacios y un numero de 5 dígitos correspondiente a un código postal. Los nombres de las calles deben poder validar calles de Madrid formadas por una sola palabra, no es necesario que reconozca "Calle Almendro Azul", porque el nombre de la calle tiene dos palabras en vez de una.

```

4 text= input("")
5 pattern = r"\b((C/|Calle)( | del | de la | De l | De La | De la | de La )([A-ZÑ][a-zñ]+),?\s+(N|n|Nº|nº)?( |)(\d+),\s+(\d{5}))\b"
6 results = re.findall(pattern, text)
7 for match in results:
8     #print(match)
9     cp = match[8]
10    n = match[7]
11    calle = match[3]
12    print(cp+'-'+calle+'-'+n)
13

```

En esta expresión regular hemos vuelto a usar `\b` por el mismo motivo explicado anteriormente. Hemos usado el indicador `|` en varias ocasiones para que haga match con distintas opciones en la forma de decir 'Calle', el resultado ha sido `(C/|Calle) ( | del | de la | Del | De La | De la | de La )`. Para detectar el nombre de la calle, hemos utilizado `([A-ZÑ][a-zñ]+)` que primero detecta una letra mayúscula, seguido de un numero indeterminado de minúsculas. Después, hemos puesto `(,)?` para reflejar que la coma es opcional usando `?`. Para el numero de la calle hemos utilizado `(N|n|Nº|nº)?( |)(\d+)`, contemplando los diferentes casos del uso de la N, volviendo a ser opcional como antes, y para el numero hemos usado `\d+` para que haga match con cualquier número. Por ultimo el código postal lo hemos representado así `(\d{5})` para usar exactamente 5 dígitos.

- Transformar cada línea a CSV extrayendo la siguiente información: Nivel de log, Hilo donde se ha producido el log (este hilo se corresponde con letras en mayúscula, minúscula y de números), Clase responsable de emitir el log y Mensaje de log

```

3 text = input()
4 pattern = r"\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\. \d{3}\s+(\w+)\s+\d+\s+---\s+[(\w+)\s+(:\w+\.)*(\w+)\s+:\s+(.*)"
5 results = re.findall(pattern, text)
6 for match in results:
7     print(''+match[0]+''+", "+match[1]+''+", "+match[2]+''+", "+match[3]+''+")

```

Para esta expresión regular, primero hemos hecho uso de `\d` para representar la fecha y la hora del log, acompañándolos de los guiones y signos de puntuación necesarios, quedando así `\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\. \d{3}`. Hemos usado `\s+` para representar un numero cualquiera de espacios.

Para representar el nivel de log hemos usado `(\w+)` para representar cualquier carácter un numero cualquiera de veces, después de una serie de espacios, números y guiones, hemos usado lo mismo de antes para representar el hilo, pero esta vez entre corchetes: `\[(\w+)\]`. ¿Para representar la clase hemos usado `(?:\w+\.)*(\w+)`, el primer grupo consiste en una serie de caracteres `(\w+)` seguidos de un punto `(\.)`, que puede haber un numero cualquiera de veces o puede no haberlo, para eso hacemos uso del asterisco, usamos `?:` para indicar que no se tenga en cuenta a ese grupo a la hora de imprimir los resultados deseados; el segundo grupo es el nombre de la clase: `(\w+)`. Por último, ponemos `\s+` para realizar los espacios en blanco antes y después de los dos puntos, y finalmente con `(.*)` nos estamos refiriendo al mensaje del log, puede ser cualquier tipo de carácter de cualquier longitud.

## PARTE 2: integración Continua

Antes de empezar, adjuntamos nuestro enlace al repositorio:

[https://github.com/miguelangel90/Metodologias\\_Practica\\_2.git](https://github.com/miguelangel90/Metodologias_Practica_2.git)

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



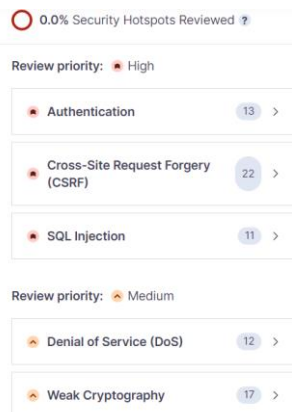
Bugs: 254 Vulnerabilidades: 3 code smells: 2k

- 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?

Tenemos las siguientes vulnerabilidades:

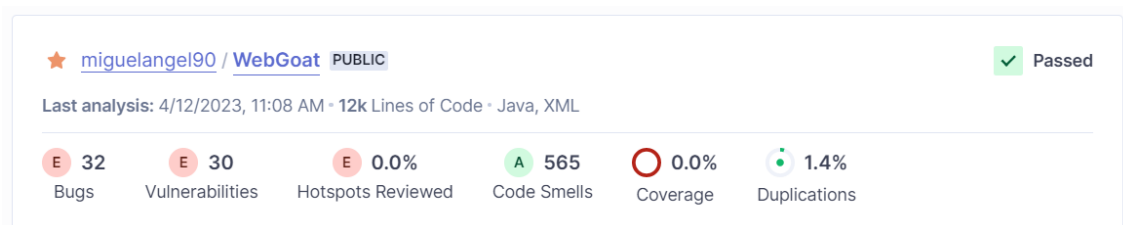
src/.../org/owasp/webgoat/container/WebSecurityConfig.java	<input type="checkbox"/> <a href="#">Don't use the default "PasswordEncoder" relying on plain-text.</a>	No tags +
	<b>Vulnerability</b> <input type="radio"/> Open <input checked="" type="radio"/> Critical <input type="radio"/> Not assigned	30min effort • 18 days ago
src/.../java/org/owasp/webgoat/webwolf/WebSecurityConfig.java	<input type="checkbox"/> <a href="#">Don't use the default "PasswordEncoder" relying on plain-text.</a>	No tags +
	<b>Vulnerability</b> <input type="radio"/> Open <input checked="" type="radio"/> Critical <input type="radio"/> Not assigned	30min effort • 18 days ago
src/.../owasp/webgoat/webwolf/mailbox/MailboxController.java	<input type="checkbox"/> <a href="#">Replace this persistent entity with a simple POJO or DTO object.</a>	No tags +
	<b>Vulnerability</b> <input type="radio"/> Open <input checked="" type="radio"/> Critical <input type="radio"/> Not assigned	10min effort • 18 days ago

## Lista de Security Hotspots:



La diferencia entre vulnerabilidades y Security Hotspots es que una vulnerabilidad es una debilidad específica que puede ser explotada, mientras que los Security Hotspots son áreas generales de preocupación en el código que pueden ser más susceptibles a errores de seguridad y, por lo tanto, a vulnerabilidades. Los Security Hotspots no son necesariamente una vulnerabilidad en sí mismos, pero son áreas donde los desarrolladores deben prestar especial atención para asegurarse de que el código sea seguro.

- 3. Haz cualquier commit para forzar una 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?



El estado del proyecto es Passed.

Esto se debe a que el autoanálisis que ha realizado SonarCloud ha determinado que no hay vulnerabilidades lo suficientemente críticas como para que pueda afectar significativamente al proyecto.

El número de vulnerabilidades no afecta a este resultado, ya que, por ejemplo, en nuestro hay 30 vulnerabilidades, pero ninguna de ellas es lo suficientemente crítica como para que pueda afectar al proyecto, en cambio, podríamos tener menos vulnerabilidades, por ejemplo, 5 vulnerabilidades, pero todas son muy críticas, y no pasan el filtro del análisis, entonces en ese caso el estado del proyecto sería Failed.

## Mitigación

1. Elige una vulnerabilidad de tipo Blocker o Critical, explica cuál es la vulnerabilidad detectada, por qué ha sido detectada (y si realmente es una vulnerabilidad y no un falso positivo).
- VULNERABILIDAD 1

src/.../org/owasp/webgoat/container/WebSecurityConfig.java

☐ [Don't use the default "PasswordEncoder" relying on plain-text.](#) No tags +

Vulnerability

Open

Critical

Not assigned

30min effort • 21 days ago

En la imagen se presenta la vulnerabilidad crítica que vamos a analizar. **“Don't use the default "PasswordEncoder" relying on plain-text”** que trata sobre el uso de contraseñas en texto plano.

La vulnerabilidad crítica que vamos a analizar consiste en que el sistema usa contraseñas en texto plano, esto es una práctica muy insegura a la hora de gestionar contraseñas, ya que cuando almacenamos contraseñas de usuarios en texto plano, es decir, sin cifrar, cualquier persona que tenga acceso a la base de datos, puede leer la información y así comprometer a la seguridad de las cuentas de los usuarios.

Un atacante que obtenga acceso no autorizado a la base de datos de contraseñas podría leer las contraseñas en texto plano y utilizarlas para acceder a las cuentas de los usuarios.

Para evitar esta vulnerabilidad crítica, es importante utilizar funciones de hash seguras y robustas para proteger las contraseñas de los usuarios. Un hash es una función que toma una cadena de entrada (en este caso, una contraseña) y la transforma en una cadena de salida que es difícil de revertir. Al almacenar las contraseñas como hashes, es posible verificar que un usuario ha ingresado la contraseña correcta sin tener que almacenar la contraseña en texto plano.

En cuanto a los falsos positivos, un falso positivo en vulnerabilidades es cuando un sistema que se encarga de la detección de vulnerabilidades dice que un sistema o aplicación tiene una vulnerabilidad cuando en realidad no es el caso. En nuestro caso, NO se trata de un falso positivo, sino que se trata de una recomendación que nos hace sonarCloud para proteger los datos, ya que estos se están almacenando sin ningún tipo de cifrado, es decir, en texto plano. Esta vulnerabilidad es muy grave, ya que como hemos dicho anteriormente, si un atacante accede a nuestra base de datos, en donde almacenamos toda la información, este podría ver toda la información sin cifrar. Es por ello que tenemos que cifrar los datos, antes de meterlos en el sistema.

- VULNERABILIDAD 2

src/.../lessons/challenges/challenge5/Assignment5.java

☐ [Change this code to not construct SQL queries directly from user-controlled data.](#) No tags +

Vulnerability

Open


Blocker

Not assigned

30min effort • 21 days ago

Esta vulnerabilidad nos dice que hay un fragmento de código en el archivo Assignment5.java, que puede ser explotado mediante una inyección SQL, ya que, en lugar de utilizar parámetros en la consulta preparada, se está concatenando directamente las variables 'username\_login' y 'password\_login' en la cadena de consulta. Esto permite que un atacante malintencionado pueda enviar valores maliciosos a través de los parámetros y manipular la consulta SQL para extraer o manipular datos de la base de datos. En este caso, es una vulnerabilidad realmente peligrosa, y no un falso positivo, esto es debido a que la concatenación de valores de entrada directamente en una cadena de consulta SQL puede permitir a un atacante manipular la consulta para extraer, modificar o eliminar datos de la base de datos

```
try (var connection = dataSource.getConnection()) {
6 PreparedStatement statement =
5 connection.prepareStatement(
4 "select password from challenge_users where userid = '"
+ username_login
+ "' and password = '"
+ 3 password_login
+ "'");
ResultSet resultSet = 7 statement.executeQuery();
}
```

 **Change this code to not construct SQL queries directly from user-controlled data.**

2. Propón una solución e impleméntela en una nueva rama del repositorio. Explica los cambios que arreglan dicha vulnerabilidad.
- VULNERABILIDAD 1


Dicha vulnerabilidad se encuentra en la siguiente ruta del proyecto:

**src/main/java/org/owasp/webgoat/container/WebSecurityConfig.java**

A continuación, vamos a mostrar donde se produce este fallo de seguridad:

```
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService);
}

}
```

 **Don't use the default "PasswordEncoder" relying on plain-text.**

Este método, se encarga de la autenticación de los usuarios en nuestro sistema, es decir, validar que un usuario es quien dice ser, para ello, vamos a pasarle un objeto **auth** de tipo **AuthenticationManagerBuilder** que se encarga de la autenticación de los usuarios.

**auth.userDetailsService(userDetailsService)** se centra en cargar los detalles de los usuarios, es decir, proporcionar a la aplicación los datos de los usuarios como el usuario, la contraseña, los roles que tiene... para autenticarlos.

A continuación, vamos a mostrar una posible solución, que básicamente consiste en meterle un método que cifre la información antes de almacenarla en la aplicación:



```

@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
    /*creamos una instancia de un objeto del codificador de contraseñas*/
    BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
    /*se está utilizando el BCryptPasswordEncoder, que es un codificador de contraseña seguro
    que utiliza la función de hash bcrypt para cifrar las contraseñas.*/
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);
}

```

En este caso, estamos usando el algoritmo de cifrado BCrypt para cifrar la información y así, evitar que toda esta se almacene en texto plano, es decir, sin cifrar en la aplicación.

Básicamente el cambio que hemos realizado es añadirle un “cifrador” de información, que en nuestro caso es BCrypt.

```

46  /*IMPORTAMOS CLASE*/
47  import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
--

```

- VULNERABILIDAD 2

Para solucionar la vulnerabilidad de la inyección SQL, hemos cambiado la consulta SQL que se encuentra dentro de la variable ‘stament’, sustituyendo las variables ‘username\_login’ y ‘password\_login’ por signos de interrogación ‘?’. Estos signos de interrogación representan los parámetros por los que van a ser sustituidos mediante la función stament.setString(). De esta forma nos aseguramos de no usar las variables directamente, usando los parámetros en la consulta preparada.

```

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();
}

```

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

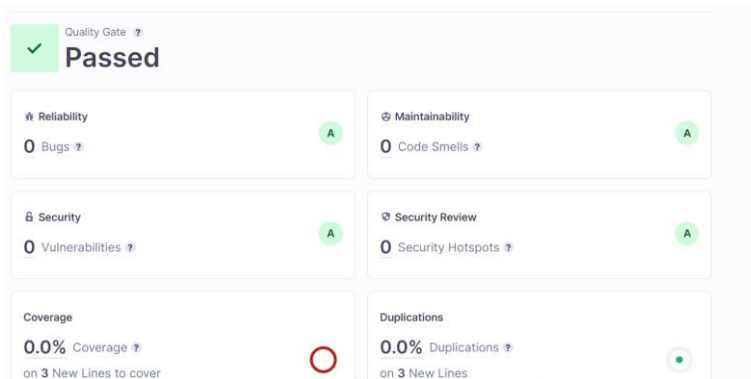
En el caso de la vulnerabilidad de inyección SQL, el Pull Request realizado pasa sin problemas en SonarCloud, sin ningún tipo de vulnerabilidad, bug, etc

2 - Update Assignment5.java

Passed

4 minutes ago

d52d9d4d

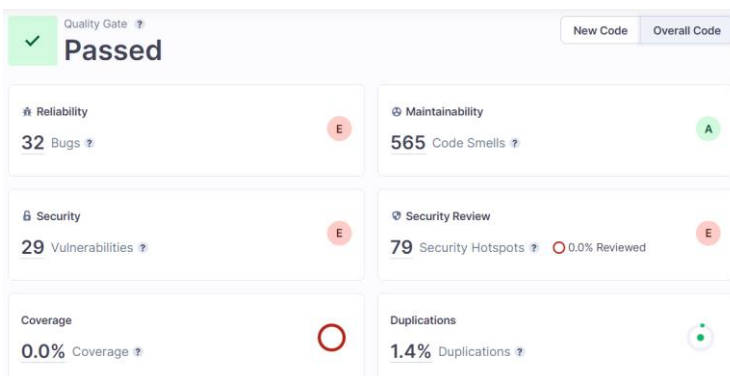


En cuanto a la vulnerabilidad del uso de contraseñas en texto plano, nuestra modificación en el código para arreglar la vulnerabilidad ha pasado el análisis de SonarCloud.



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 realizar el merge de la rama de la vulnerabilidad de inyección SQL, el único cambio que se encuentra en el proyecto es el número de vulnerabilidades, que pasa de 30 a 29, por lo tanto, esto quiere decir que hemos solucionado la vulnerabilidad correctamente.



Por último, tras realizar el merge de la rama de la vulnerabilidad del uso de contraseñas en texto claro, hemos visto reducido en una unidad el número de vulnerabilidades, ya que pasó de 29 a 28, es decir, vulnerabilidad corregida, pero SonarCloud al volver a realizar el análisis de nuestro proyecto, no lo aprueba, ya que debe haber alguna regla que nuestro sistema no pasa, por lo tanto, da como resultado **failed**.

