




2022/2023

PRÁCTICA 3

METODOLOGÍA DE DESARROLLO SEGURO

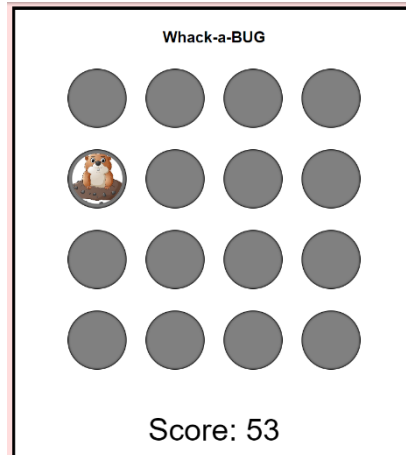
JOSÉ LUIS MEZQUITA JIMÉNEZ Y MIGUEL ANGEL VILLANUEVA
INGENIERIA DE LA CIBERSEGURIDAD
2 CUATRIMESTRE



WHACK-A-MOLE

Este juego es el famoso juego que consiste en derribar los topos de cada circulo, para derribar los topos y conseguir puntos, basta con pulsar sobre ellos, pero hay que tener en cuenta que los topos van apareciendo y desapareciendo constantemente de forma aleatoria.

Para automatizarlo vamos a usar la herramienta de automatización SELENIUM en donde vamos a hacer click sobre todos los botones todo el rato, es decir, es como si golpeases todos los agujeros haya o no haya un topo y así hacerlo hasta que se llegue a la cifra de 10.000 puntos en donde una vez que lleguemos, obtendremos la flag.



A continuación, vamos a mostrar el código que hemos desarrollado para conseguir este reto y así obtener la flag.

En esta captura, se muestra nuestras librerías importadas. Después vamos a cargar el buscador que hemos elegido, que en nuestro caso es Chrome, en la variable driver. En la siguiente línea, hemos hecho un get de nuestro html proporcionado en el enunciado, que básicamente nos va a abrir el html en Chrome. La variable contador=0 la usaremos para llevar un recuento de la puntuación acumulada y por último, tenemos un time.sleep(5) en donde vamos a esperar 5 segundos para que se cargue el html correctamente

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
driver= webdriver.Chrome()
driver.get("C:\\Users\\USUARIO\\Desktop\\URJC\\3º Ciberseguridad\\2º CUATRIMESTRE\\METODOLOGIA DESARROLLO SEGURO\\PRACTICA 3\\TOPOS\\index.html")
contador= 0
time.sleep(5)
```

En la siguiente captura, se aprecia el bucle que hemos usado, cuya condición es hasta que lleguemos a la puntuación de 10.000 topos cazados, en donde parara de ejecutar el bucle.

Hole, en la siguiente línea, es una lista, en donde hemos buscado los elementos que tengan por nombre de clase "hole" que corresponden con los agujeros de los topos. En las siguientes 16 líneas, estamos clickando a la vez en todas para que cuando aparezca un topo, lo coja, de tal manera que no se va a escapar ninguno, debido a que vamos a estar pulsando constantemente en todos.

La variable actual, va a buscar el elemento por id= score que va a corresponder con la puntuación, una vez tengamos localizado esto, vamos a extraer la información usando la función text, en donde el valor

que se muestra en la web, nos lo devuelve como string, por lo que hacemos la transformación a entero para sumar y llegar a 10.000.

El input colocado al final lo usamos para quedar congelado el programa y que no avance ya que tuvimos problemas al principio ya que cuando ejecutaba y llegaba a la puntuación, se cerraba y no nos daba tiempo a coger la flag.

```
while contador < 10000:
    hole = driver.find_elements(By.CLASS_NAME, "hole")
    hole[0].click()
    hole[1].click()
    hole[2].click()
    hole[3].click()
    hole[4].click()
    hole[5].click()
    hole[6].click()
    hole[7].click()
    hole[8].click()
    hole[9].click()
    hole[10].click()
    hole[11].click()
    hole[12].click()
    hole[13].click()
    hole[14].click()
    hole[15].click()

    actual= driver.find_element(By.ID, 'score')
    contador=int(actual.text)

input()
driver.close()
```

[illegible]

10-FINGERS



Este juego consiste en ver la velocidad de escritura que tenemos con nuestros dedos, para ello, se nos pide escribir 20 palabras en menos de 5 segundos, lo cual aparenta ser imposible, pero para nosotros no, ya que vamos a usar la herramienta SELENIUM para automatizar la escritura.

Así una explicación por encima de como lo hemos desarrollado es, localizas la primera palabra, la obtienes con la función text, localizas el cuadrado y pegas la palabra.

Para empezar, igual que en el ejercicio anterior, cargamos el buscador que vamos a elegir, que en nuestro caso es Chrome, después, vamos a abrir con el buscador el html que se nos ha proporcionado. En la siguiente línea, hacemos una espera de 10 segundos para que se cargue correctamente el html y no tengamos ningún problema.

En elemento vamos a tener localizada la primera palabra, gracias a la función fin_element, en donde vamos a buscar por nombre de clase. En realidad, da igual con lo que busques, sea CSS, id, nombre de clase, XPATH... el único problema que puede haber es a la hora de que haya repetidos que siempre nos va a coger el primero. Recuadro va a tener localizado el elemento del cuadrado que va a ser el lugar donde vamos a escribir la palabra.

```
# importamos las librerías de selenium
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

driver = webdriver.Chrome()
driver.get("C:\\Users\\USUARIO\\Desktop\\URJC\\3º Ciberseguridad\\2º CUATRIMESTRE\\METODOLOGIA DESARROLLO SEGURO\\PRACTICA 3\\FINGERS\\index.html")
wait = WebDriverWait(driver, 10)

# localizamos la primera palabra
elemento = driver.find_element(By.CLASS_NAME, "current")

# localizamos le recuadro
recuadro = driver.find_element(By.ID, "textInput")
```

Ahora vamos a hacer un bucle de 20 iteraciones que son el número de palabras que hay que escribir, para ello sacamos la primera palabra, y la metemos la palabra en el recuadro y a continuación un espacio para que nos muestre la siguiente palabra. Después de esto localizamos la siguiente palabra, ya que esto va avanzando y volvemos a hacer lo mismo.

```
for i in range(20):
    # sacamos la primera palabra
    word = elemento.text

    # metemos la palabra
    recuadro.send_keys(word)
    recuadro.send_keys(Keys.SPACE)

    # nos colocamos sobre la nueva palabra
    elemento = driver.find_element(By.CLASS_NAME, "current")
```

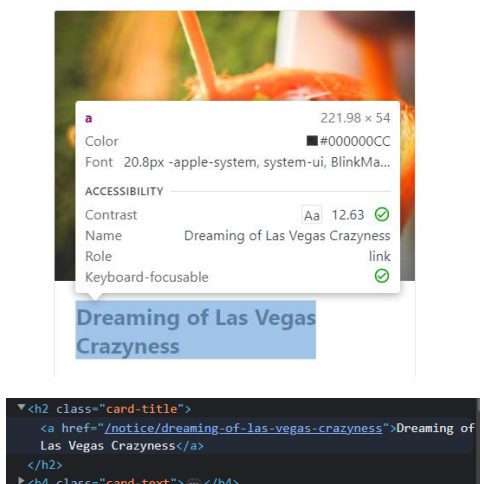
BLOG

En este ejercicio se nos muestra un blog normal, para resolver el ejercicio, tenemos que buscar todas las veces que aparece la cadena de caracteres "URJC", solo esta, todo en mayúsculas y sola, es decir, sin estar contenida en otro string como, por ejemplo: "holaURJCadios".

Este blog estaba formado por una página principal de la cual salen enlaces a otras páginas, por lo tanto, para realizar el ejercicio, así a modo resumen, teníamos dos opciones, la primera era descargarnos (hacer una copia de todo el html) la página al completo y hacer una búsqueda de manera recursiva y aplicar un comando grep que localizase nuestro string y luego con un wc -l contar todas las veces que aparece, y la otra que es la que hemos usado que consiste en usar SELENIUM, en donde hemos seguido un procedimiento similar, que consistía en recorrernos todas las webs, clicando sobre los enlaces, obtener el html y con una función "contar" hemos usado una expresión regular para detectar el número de apariciones por página de nuestro string y así contarlas y añadirlas a un contador.

En la siguiente captura se muestran las librerías que hemos usado y las dos funciones que nos hemos creado, la primera función, nos consigue todos los links de esa web. En donde para ello, nos creamos un conjunto donde vamos a almacenar todos los enlaces, y vamos a recorrernos todos los elementos que tengan como nombre de etiqueta en el html "a", después vamos a usar la función conseguir atributo en donde vamos a obtener los atributos que empiecen por "href" que son los que se usan para los enlaces. Comprobamos que sea un link, es decir, que no sea valor False, y metemos en nuestro conjunto.

Por otro lado, la segunda función va a contar las veces que aparece en un texto URJC es por ello por lo que hemos usado una expresión lógica, en donde hemos retornado el valor de matches (len)



```
import re
from selenium import webdriver
from selenium.webdriver.common.by import By

def get_all_links(driver):
    links = set()
    for a in driver.find_elements(By.TAG_NAME, 'a'):
        link = a.get_attribute('href')
        if link:
            links.add(link)
    return links

def count_occurrences(string, substring):
    pattern = r"\bURJC\b"
    matches = re.findall(pattern, string)
    return len(matches)
```

En esta captura, hemos hecho lo de siempre, para iniciar nuestra web, cargar el buscador en la variable driver y abrirlo con la url del blog, además, nos hemos creado un conjunto en donde vamos a ir almacenando todos los enlaces que vayamos obteniendo en cada visita a una página.

```
##### MAIN PROGRAM #####
driver = webdriver.Chrome()
visited_links = set()
all_links = set()
# Visitar la página principal
driver.get('https://r2-ctf-vulnerable.numa.host/')
# Obtener todos los enlaces de la página principal
all_links = get_all_links(driver)
```

En esta captura, se muestra el código de nuestro programa, en donde, como se puede apreciar, tenemos un while, que se va a repetir hasta que el conjunto que almacena los enlaces quede totalmente vacío. En este bucle, vamos a recorrer recursivamente todos los enlaces, sin repetición. Para ello, sacamos el primer enlace con un pop, vemos que empieza con la url indicada en la función "startswith", que sea un link, que no esté visitado y que no termine con el carácter #. Si esta condición se cumple, nos metemos en el if, en donde vamos a imprimir por pantalla el link que estamos visitando, una vez mostrado por pantalla, capturamos todos los links de esa página, y los recorremos (child link), en donde comprobamos que realmente sea un link, y si se da el caso que lo es, lo metemos en nuestro conjunto de links.

Una vez terminado todo esto, nos vamos a dedicar a contar las veces en las que aparece "URJC", en donde, vamos a recorrer todos los links almacenados que tenemos, y uno a uno vamos a ver si hay strings "URJC".

Para ello, cargamos el enlace, obtenemos la página, en donde nos devuelve todo el código html, una vez esto, se lo pasamos a nuestra función contar, donde se empleará la expresión regular y nos devolverá el número de apariciones.

```
# Recorrer recursivamente todos los enlaces de la página principal y sus enlaces hijos
while all_links:
    link = all_links.pop()
    if link.startswith('https://r2-ctf-vulnerable.numa.host/') and link not in visited_links and not link.endswith('#'):
        visited_links.add(link)
        driver.get(link)
        print(f"Visitando: {link}")
        child_links = get_all_links(driver)
        for child_link in child_links:
            if child_link.startswith('https://r2-ctf-vulnerable.numa.host/') and child_link not in visited_links and not child_link.endswith('#'):
                all_links.add(child_link)

# Contar las apariciones de "URJC" en todas las páginas visitadas
count = 0
for link in visited_links:
    driver.get(link)
    page_source = driver.page_source
    count += count_occurrences(page_source, 'URJC')
print(f"El número de apariciones de 'URJC' en todas las páginas visitadas es: {count}")

# Cerrar el navegador
driver.quit()
```

Este ha sido el resultado de nuestro programa, en donde, como vemos, como manera de depurar, hemos decidido imprimir todas las webs que iba nuestro programa recursivo visitando:

```
Visitando: https://r2-ctf-vulnerable.numa.host/notice/when-some-where-who-people
Visitando: https://r2-ctf-vulnerable.numa.host/notice/down-been-said-they-use-could-would-long
Visitando: https://r2-ctf-vulnerable.numa.host/notice/of-she-did-day-more-did-into
Visitando: https://r2-ctf-vulnerable.numa.host/notice/see-other-two-the-you-have
Visitando: https://r2-ctf-vulnerable.numa.host/notice/more-him-could-with-them-a-no-was-made-my
Visitando: https://r2-ctf-vulnerable.numa.host/notice/with-like-that-about-could-call-many-is
Visitando: https://r2-ctf-vulnerable.numa.host/notice/will-made-your-about-may
El número de apariciones de 'URJC' en todas las páginas visitadas es: 265
```

AGENDA en C

En este reto nos proporcionan un código en C que simula el uso de una agenda en la que podemos añadir nuestros contactos y consultarlos, estando cada contacto en una posición entre 0 y 9, ambos incluidos. El fallo de este código, se encuentra en que no controla que la posición introducida sea negativa, es por ello, que la flag esta en una de ellas (6).

El código se compone de la función main, que va imprimiendo mensajes que indican lo que podemos hacer; y dos funciones, add_contact () y show_contact (), que sirven para añadir y visualizar contactos respectivamente. Las funciones main y add_contact no contienen nada interesante que se pueda explotar, ya se comprueba en cada momento que la posición o opción que introducimos está dentro del rango permitido. Sin embargo, si nos fijamos, en la función show_contact, solo comprueba que la posición introducida es mayor que el tamaño de la agenda, es decir, que comprueba si la posición introducida es mayor que la posición más alta permitida, en este caso 9, pero no comprueba si la posición introducida es menor de lo permitido, en este caso 0, por lo que se pueden introducir posiciones negativas.

```
void show_contact(){
    printf("Que contacto quieres recuperar? Indica su posicion en la agenda.\n");
    int pos;
    scanf("%d", &pos);

    if(pos ≥ SIZE){
        printf("Posicion invalida! Que tramas?\n");
        return;
    }

    printf("El numero de la posicion %d es %s\n", pos, agenda[pos]);
}
```

Si vamos comprobando una a una, cada posición negativa, vemos que nos va devolviendo un string "null" o caracteres extraños, hasta que llegamos a la posición -6, en la que se encuentra la flag, que en este caso es: URJC{Why_so_negative}

```

El numero de la posicion -4 es ♦(♦♦
Bienvenido a la agenda. Hay 10 huecos disponibles para almacenar contactos.
Que deseas hacer?
1.- Anadir contacto en la agenda
2.- Leer contacto de la agenda
3.- Salir
2
Que contacto quieres recuperar? Indica su posicion en la agenda.
-5
El numero de la posicion -5 es (null)
Bienvenido a la agenda. Hay 10 huecos disponibles para almacenar contactos.
Que deseas hacer?
1.- Anadir contacto en la agenda
2.- Leer contacto de la agenda
3.- Salir
2
Que contacto quieres recuperar? Indica su posicion en la agenda.
-6
El numero de la posicion -6 es URJC{Why_so_negative}
Bienvenido a la agenda. Hay 10 huecos disponibles para almacenar contactos.
Que deseas hacer?
1.- Anadir contacto en la agenda
2.- Leer contacto de la agenda
3.- Salir

```

Esta vulnerabilidad es bastante sencilla de arreglar, bastaría con comprobar que la posición introducida no puede ser menor que 0 en la función `show_contact`, tal y como se hace en la función `add_contact`, para así evitar acceder a posiciones negativas. Quedaría de la siguiente forma:

```

void show_contact(){
    printf("Que contacto quieres recuperar? Indica su posicion en la agenda.\n");
    int pos;
    scanf("%d", &pos);

    if(pos ≥ SIZE || pos < 0){
        printf("Posicion invalida! Que tramas?\n");
        return;
    }

    printf("El numero de la posicion %d es %s\n", pos, agenda[pos]);
}

```


AGENDA EN PYTHON

En este reto nos proporcionan un código en Python que simula el uso de una agenda en la que podemos añadir nuestros contactos y consultarlos, estando cada contacto en una posición entre 0 y 9, ambos incluidos.

El código es prácticamente igual al código en C del reto anterior, utiliza las mismas funciones, idénticas al código en C, solo que esta vez no hay nada almacenado en las posiciones negativas.

Pero esta agenda tiene algo que la distingue y es que el código importa la librería 'os', esta librería no es utilizada en ninguna parte del código. Buscando información acerca de esta librería, nos permite acceder a funcionalidades específicas del sistema operativo, como la configuración de variables de entorno. Si echamos un vistazo al archivo docker_compose.yml, podemos observar que hay una variable de entorno llamada CHALLENGE_FLAG=URJC {change_me}. El fallo está en que en un lugar donde debería ir un entero, podemos meter un string, en este caso la variable de entorno.

```
import os

# Te recuerda a algun
SIZE = 10
agenda = [""] * SIZE

environment:
  - CHALLENGE_FLAG=URJC{change_me}
```

Al tener la variable de entorno que puede contener la flag, la librería para acceder a la variable de entorno, y viendo que la agenda es un array de strings, podemos usar la función environ.get de la librería os para acceder al contenido de la variable de entorno. Por lo tanto, cuando accedemos a la opción de leer contacto de la agenda, si introducimos la función ya mencionada, nos salta un error, pero conseguimos obtener la flag, que en este caso es: URJC{Python2_is_old}.

```
(migueltangel90@kali)-[~]
$ nc vulnerable.numa.host 9994
Bienvenido a la agenda. Hay 10 huecos disponibles para almacenar contactos.
Que desees hacer?
1.- Anadir contacto en la agenda
2.- Leer contacto de la agenda
3.- Salir

2
Que contacto quieres recuperar? Indica su posicion en la agenda.

os.environ.get('CHALLENGE_FLAG')
Traceback (most recent call last):
  File "/challenge.py", line 56, in <module>
    main()
  File "/challenge.py", line 48, in main
    show_contact()
  File "/challenge.py", line 24, in show_contact
    pos = int(input())
ValueError: invalid literal for int() with base 10: 'URJC{Python2_is_old}'
```

Para evitar esta vulnerabilidad, bastaría con eliminar la línea de código en la que se importa la librería os, otra opción podría ser

comprobar el contenido de lo que se introduce y ver si contiene el nombre de la variable de entorno que no queremos que se use, mediante una sentencia if se podría conseguir esto, terminando la ejecución del programa si se cumple la condición.

BITCOIN

En este reto nos proporcionan un código en C en el que tenemos un numero inicial de bitcoins, 100 en este caso, tenemos 2 opciones, gastarlos o recargarlos, mediante la función gastar y recargar respectivamente. Para comprar la flag necesitamos 150 bitcoins. Para resolverlo, tenemos que ver el fallo que hay a la hora de introducir el número, ya que analizando el resto de ficheros/variables del SS00 no hemos encontrado nada, para ello, hemos detectado que el fallo en la implementación está a la hora de introducir el ultimo entero, ya que si metes el máximo número entero que se puede almacenar, puedes sacar la flag.

El problema viene cuando vemos que la función recargar simplemente imprime un mensaje por pantalla y no modifica el número de bitcoins.

```
void recargar(){
    printf("De acuerdo, envíame tu número de tarjeta, la fecha de caducidad y el número que sale por la parte de atrás por favor. Tranquilo, te puedes fiar de mí... o mejor no, sigue buscando :)");
    return;
}
```

En la función gastar podemos gastar los bitcoins en diferentes cosas, pero la única que importa es la flag ya que devuelve mediante la función getenv la flag contenida en la variable de entorno CHALLENGE_FLAG.

```
}
}else if(gastoEn==3){
    printf("Cuántas flags quieres comprar?\n");
    int cuantas;
    scanf("%d", &cuantas);
    if(cuantas <= 0){
        printf("Introduce una cantidad positiva diferente a 0\n");
        return;
    }
    if(PRESUPUESTO >= 150*cuantas){
        PRESUPUESTO = PRESUPUESTO - 150*cuantas;
        printf("\n\n%s\n", getenv("CHALLENGE_FLAG"));
        *(int*)0 = 0; // Bye bye
    }else{
        printf("Te toca ahorrar crack!\n");
        return;
    }
}
```

Como no hay ninguna forma de aumentar la cantidad de bitcoins, tenemos que encontrar una manera de que nuestro presupuesto sea mayor que el precio total de las flags. Esto podría hacerse intentando hacer un overflow, de forma que multiplicando dos números positivos demasiado grandes de como resultado un número negativo. Buscando información acerca de cual es el mayor número entero de tipo int que puede interpretar C, tenemos que ese número es 2147483647. Por lo tanto, si introducimos este número podemos obtener la flag.

```

(miguelangel90@kali)-[~]
$ nc vulnerable.numa.host 9992

Bienvenido a la nueva criptotienda. Que deseas hacer?
1.- Gastar Bitcoins
2.- Recargar Bitcoins
3.- Salir
Tu saldo actual es: 100
Opcion: 1
En que quieres gastar tus bitcoins?
1.- Cerveza 2$
2.- Barcos y pujas 4$
3.- Flags (Quiero aprobar!) 150$
Tu saldo actual es: 100
Opcion: 3
Cuantas flags quieres comprar?
2147483647

URJC{Ojala_funcionara_con_el_banco}

```

En este caso la flag es URJC{Ojala_funcionara_con_el_banco}. Para que no se pueda explotar esta vulnerabilidad, se podría poner un número máximo de las flags que se quieren comprar, así no se pueden poner números enormes evitando así el overflow.

CALCULADORA

En este reto se nos proporciona un código en Java que simula el uso de una calculadora mediante el uso de funciones como multiplicar, dividir, etc. Nos piden una serie de requisitos que tiene que cumplir esta calculadora, así que iremos uno por uno explicándolos.

-Debe protegerse contra Overflows de forma correcta: por ejemplo, la multiplicación de dos números positivos nunca puede dar un número negativo. En caso de que sea posible overflow, tirar ArithmeticException.

Para esto hemos modificado el método de multiplicar:

```

public long multiply(long a, long b){
    log( message: "Multiply %s * %s", a, b);
    long result = Math.multiplyExact(a,b);
    return result;
}

```

Para que el resultado de multiplicar dos positivos de un número negativo al menos uno de los dos números tiene que ser bastante grande, por eso el método devuelve un Long, para que pueda interactuar con números mas grandes. Pero esto no vale, por lo tanto, usamos la función multiplyExact, esta función multiplica los números que le pasamos, y en caso de que ocurra, lanza automáticamente una ArithmeticException. Hemos usado los siguientes test para comprobar el funcionamiento:

```

@Test
public void multiply(){
    SecureCalculator calculator = new SecureCalculator(log);
    Assertions.assertEquals( expected: 60, calculator.multiply( a: 20, b: 3));
}

@Test
public void overflow(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertThrows(java.lang.ArithmeticException.class, () -> calculator.multiply(Integer.MAX_VALUE, 2));
}

```

- En caso de intento de división entre 0, debe tirar `ArithmeticException`.

Para resolver esto, hemos comprobado si el divisor es igual a 0, y en caso de que lo sea, lance la excepción; si no es 0, realiza la división correctamente.

```

public double divide(double a, double b){
    log( message: "Divide %s / %s", a, b);
    if (b == 0){
        throw new ArithmeticException();
    }else{
        double n = a / b;
        return n;
    }
}

```

Hemos usado los siguientes test para comprobar el funcionamiento:

```

@Test
public void division(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertEquals( expected: 2, calculator.divide( a: 6, b: 3));
}

@Test
public void divisionEntreCero(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertThrows(java.lang.ArithmeticException.class, () -> calculator.divide( a: 2, b: 0));
}

```

- Debe generar números aleatorios de forma correcta, en el límite pedido

Para resolver esto, hemos usado la función `Math.random` y lo hemos multiplicado por el número límite que queremos que se genere, pero lo importante es la conversión a un número entero, ya que esta función devuelve un número de tipo `double`, y queremos que sea un entero.

```

public int getRandomNumber(){
    log( message: "Generating rnd with bound MAX VALUE");
    return getRandomNumber(Integer.MAX_VALUE);
}

/**
 * Safely generate unique numbers, always less than bound
 * @return random number in range [0, bound)
 */
public int getRandomNumber(int bound){
    log( message: "Generating rnd with bound %s", bound);
    int random = (int) (Math.random() * bound);
    return random;
}

```

Hemos usado los siguientes test para comprobar el funcionamiento:

```

@Test
public void random_bound(){
    SecureCalculator calculator = new SecureCalculator();
    int min = 0;
    int max = 10;
    int n = calculator.getRandomNumber(max);
    Assertions.assertTrue( condition: n >= min && n < max);
}

@Test
public void random(){
    SecureCalculator calculator = new SecureCalculator();
    int min = 0;
    int max = Integer.MAX_VALUE;
    int n = calculator.getRandomNumber();
    Assertions.assertTrue( condition: n >= min && n < max);
}

```

- Debe detectar correctamente cuando un número es par y cuando un número es impar

Para poder comprobar esto, primero hemos hecho uso de la función mod que calcula el módulo entre dos números.

```
public int mod(int a, int b){  
    log( message: "%s mod %s", a, b);  
    int res = a % b;  
    return res;  
}
```

Para ver si es par, simplemente usamos la función mod para hacer el modulo del numero que queremos entre 2, y comprobamos si el resultado es 0.

```
public boolean isEven(int a){  
    log( message: "%s is even", a);  
    int b = 2;  
    return mod(a, b) == 0;  
}
```

Para ver si es impar, hemos hecho lo mismo, pero teniendo en cuenta que, si el número que queremos comprobar es un número negativo, la función mod devuelve un -1 y no un 1.

```
public boolean isOdd(int a){  
    log( message: "%s is odd", a);  
    int b = 2;  
    if (mod(a,b)==1 || mod(a,b)==-1){  
        return true;  
    }else {  
        return false;  
    }  
}
```

Hemos usado los siguientes test para comprobar el funcionamiento:

```

@Test
public void mod(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertEquals( expected: 1, calculator.mod( a: 5, b: 2));
}

@Test
public void esPar(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertEquals( expected: true, calculator.isEven( a: 2));
    Assertions.assertEquals( expected: false, calculator.isEven( a: 212121235));
}

@Test
public void esImpar(){
    SecureCalculator calculator = new SecureCalculator();
    Assertions.assertEquals( expected: true, calculator.isOdd( a: -9));
    Assertions.assertEquals( expected: false, calculator.isOdd( a: 212121238));
}

```

- Debe emitir mensajes de log si ha sido configurada para ello.

Para asegurarnos de que se emite siempre un mensaje de log, en cada función hemos añadido un log al principio de cada función, como se pueden ver en las imágenes anteriores.

Hemos usado el siguiente test para comprobar el funcionamiento:

```

@Test
void verifyLogger(){
    Logger mockLog = mock(Logger.class);
    SecureCalculator calculator = new SecureCalculator(mockLog);

    calculator.divide( a: 8, b: 1);
    verify(mockLog).info(anyString());

    calculator.multiply( a: 2, b: 3);
    verify(mockLog, times( wantedNumberOfInvocations: 2)).info(anyString());

    calculator.mod( a: 5, b: 2);
    verify(mockLog, times( wantedNumberOfInvocations: 3)).info(anyString());

    calculator.getRandomNumber( bound: 10);
    verify(mockLog, times( wantedNumberOfInvocations: 4)).info(anyString());
}

```

```

calculator.isEven( a: 6);
verify(mockLog, times( wantedNumberOfInvocations: 6)).info(anyString());

calculator.isOdd( a: 7);
verify(mockLog, times( wantedNumberOfInvocations: 8)).info(anyString());

calculator.getRandomNumber();
verify(mockLog, times( wantedNumberOfInvocations: 10)).info(anyString());

calculator.multiply(Integer.MAX_VALUE, b: 2);
verify(mockLog, times( wantedNumberOfInvocations: 12)).info(anyString());

verifyNoMoreInteractions(mockLog);

```

El resultado es el siguiente:

Flag: URJC{Ahora_implementate_unas_derivadas}

Resultado de la evaluacion

```

[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.example:unit-tests >-----
[INFO] Building unit-tests 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ unit-t
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resourc
[WARNING] Using platform encoding (UTF-8 actually) to copy filte
[INFO] skip non existing resourceDirectory /tmp/495e6a27da30266a
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encodin
[INFO] Compiling 1 source file to /tmp/495e6a27da30266a02fb38136
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-tes
[WARNING] Using platform encoding (UTF-8 actually) to copy filte
[INFO] skip non existing resourceDirectory /tmp/495e6a27da30266a

```


OPTIMIZER

Para realizar este reto nos proporciona un proyecto en Java con bastantes archivos como para ir uno a uno analizándolos, por lo tanto, hemos hecho uso de SonarCloud, para que al igual que en la practica anterior, nos pueda decir las vulnerabilidades y fallos que tiene el proyecto.

Al analizarlo, nos dice que no encuentra vulnerabilidades como tal, pero si que tiene varios security Hotspots, entre ellos destaca uno que tiene que ver con la autenticación.

José Luis > OPTIMIZER > main

Summary Issues **Security Hotspots** Measures Code Activity

0.0% Security Hotspots Reviewed

To review Fixed Safe

22 Security Hotspots to review

Review priority: High

- Authentication (1)

Review this hard-coded URL, which may contain a password.

Review priority: Medium

- Weak Cryptography (2)

Review priority: Low

- Others (19)

22 of 22 shown

Review this hard-coded URL, which may contain a password.

Hard-coded passwords are security-sensitive [java:S2068](#)

Status: To Review
This Security Hotspot needs to be reviewed to assess whether the code poses a risk.

Where is the risk? What's the risk? Assess the risk How to fix it? Activity

core/.../main/java/es/urjc/etsii/grafu/util/StringUtil.java

```
35     var result = decoder.decode(encodedBytes);
36     return new String(result, charset);
37 }
38
39 // Mis credenciales de acceso a la API
40 public static final String API_KEY =
    "http://miralabase:VVJK03tNZWpvc19wb25lcmxvX2VuX0dpdGh1Yl9TZWNvZXR9@192.168.10.1/";
41
42 /**
43  * Base 64 encode a string. Uses UTF-8
44  *
45  * @param s String to encode
```

Puede parecer que no hay nada raro, pero si nos fijamos, hay un enlace sospechoso.

core/.../main/java/es/urjc/etsii/grafu/util/StringUtil.java

Show 34 more lines

```
35     var result = decoder.decode(encodedBytes);
36     return new String(result, charset);
37 }
38
39 // Mis credenciales de acceso a la API
40 public static final String API_KEY =
    "http://miralabase:VVJK03tNZWpvc19wb25lcmxvX2VuX0dpdGh1Yl9TZWNvZXR9@192.168.10.1/";
41
42 /**
43  * Base 64 encode a string. Uses UTF-8
44  *
45  * @param s String to encode
```

Review this hard-coded URL, which may contain a password.

Como bien dice, si miramos la base, en este caso Base 64, podemos encontrar la flag

Input

```
VVJKQ3tNZWpvc19wb25lcmxvX2VuX0dpdGh1Y19TZWNyZXRS
```

Output

```
URJC{Mejor_ponerlo_en_Github_Secret}
```

Y con esto conseguimos la flag.