Practica 11 - Grupo 06 - Auditoría de seguridad

Jaime Antolín, Álvar Domingo, Pablo Jurado, Leire Osés y Sans Undertale declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con nadie. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.

Vulnerabilidad 1

Ruta(s) de la aplicación involucrada(s)
127.0.0.1:5000/show_question?id=3

Tipo de vulnerabilidad
XSS Persistente

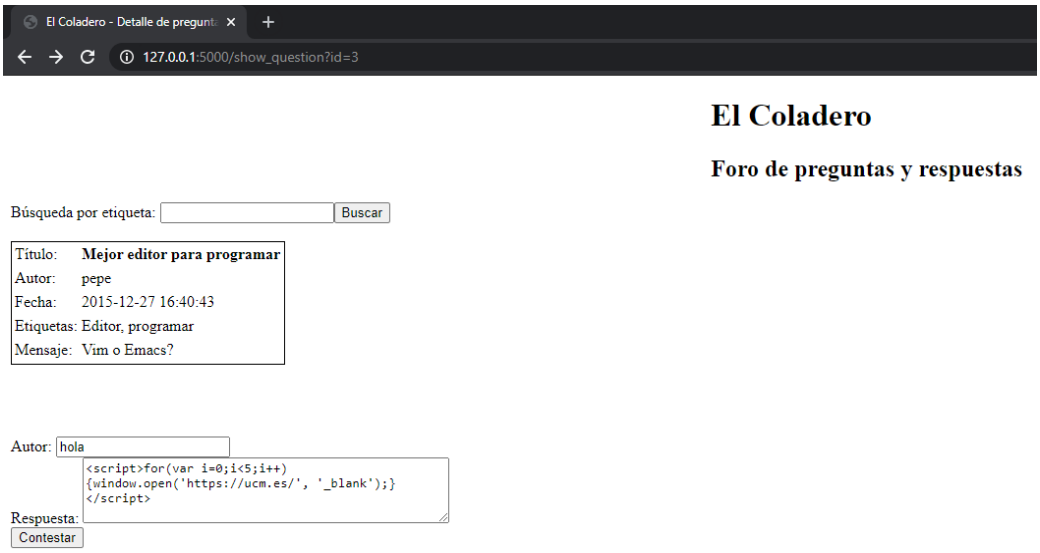
Causante de la vulnerabilidad:
Introducimos código en javascript entre etiquetas HTML como por ejemplo <script> para modificar el comportamiento deseado de la web.

Situaciones peligrosas o no deseadas que puede provocar:

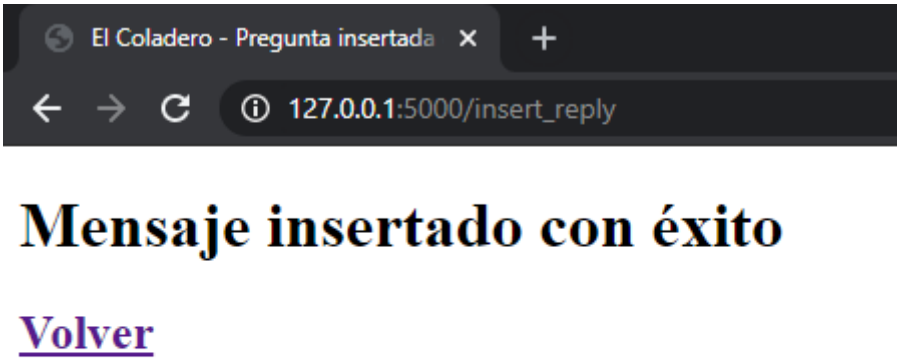
- **El script puede robarnos las cookies**
- **Obtener el ID de sesión**
- **Modificar el DOM de nuestra página web**

Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):

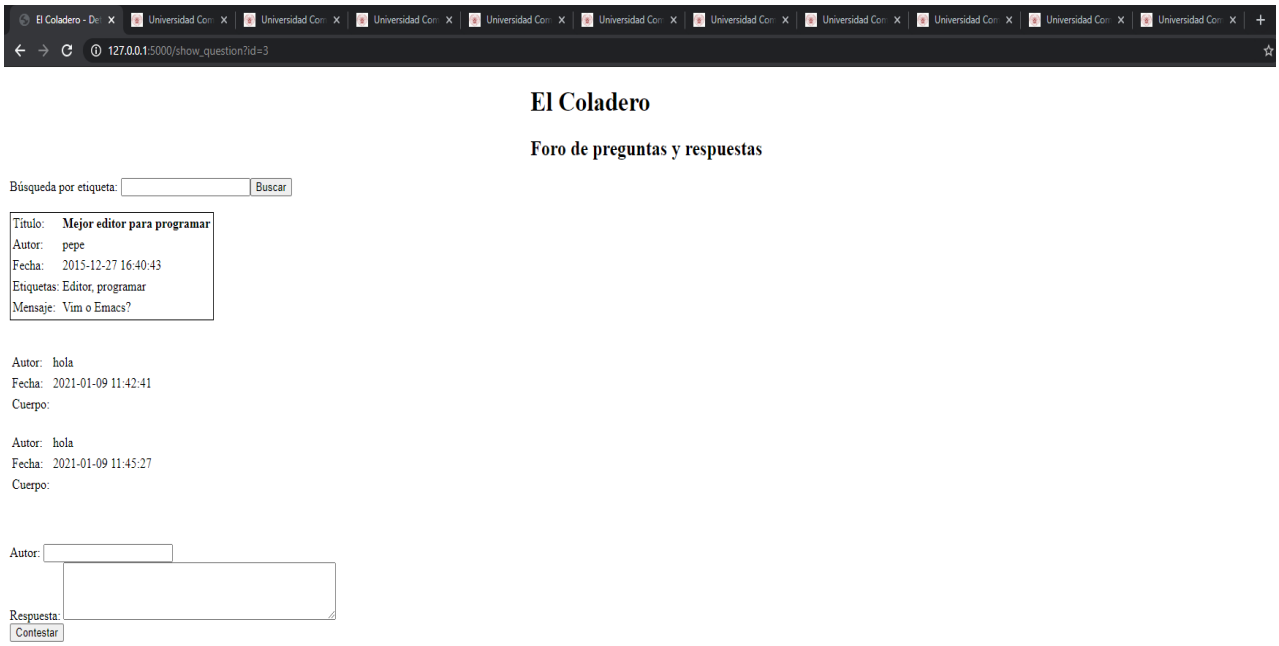
1) **Insertamos el script malicioso como respuesta a un hilo**



2) **Confirmamos que se ha insertado**



3) **Al apretar en volver, vemos que se ejecuta nuestro script (en este caso dos veces porque lo inserte dos veces) y el contenido de nuestras respuestas están vacíos**



Medidas para mitigar la vulnerabilidad

Si utilizamos la función replace en el cuerpo del mensaje donde previamente hemos insertado el script, y reemplazamos las aperturas de etiquetas potencialmente maliciosas por códigos HTML, evitaremos los ataques, ya que podrá interpretar los caracteres interpretados y los mostrará en la página en vez de ejecutar el código. Para mitigar también otras vulnerabilidades se puede usar la función html.escape(), con carácter más general.

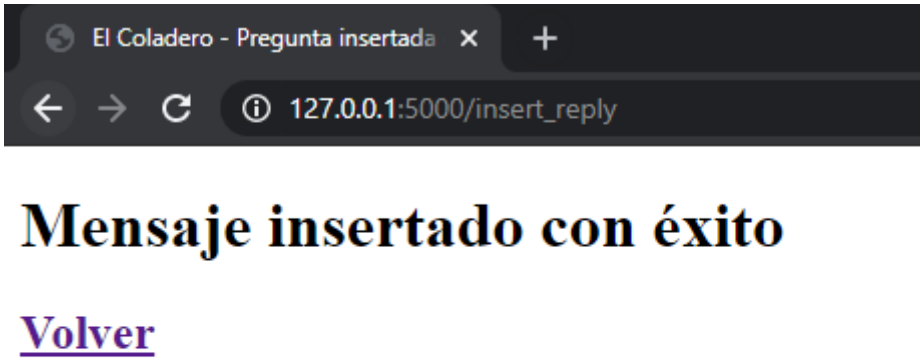
```python
@app.route('/insert_reply', methods=['POST'])
def insert_reply():
    author = request.form['author']
    body = request.form['body']
    question_id = request.form['question_id']

    body = body.replace("<", "&#60;")
    body = body.replace(">", "&#62;")

    conn = sqlite3.connect(DBPATH)
    cur = conn.cursor()
    qbody = """INSERT INTO Replies(author,body,time,question_id)
              VALUES (:author, :body, CURRENT_TIMESTAMP, :question_id)"""
    params = {'author': author, 'body': body, 'question_id': question_id}
    cur.execute(qbody, params)
    conn.commit()
    conn.close()
    return render_template("insert_ok.html", url=url_for("show_question", id=question_id))
```

1) **Ahora añadimos el mismo código**

2) **Se vuelve a añadir correctamente**



3) **Y ahora al volver no se ejecuta ningún script, y podemos ver el texto introducido en el cuerpo de la entrada**

Vulnerabilidad 2

Ruta(s) de la aplicación involucrada(s)
127.0.0.1:5000/search_question

Tipo de vulnerabilidad
XSS Reflejado

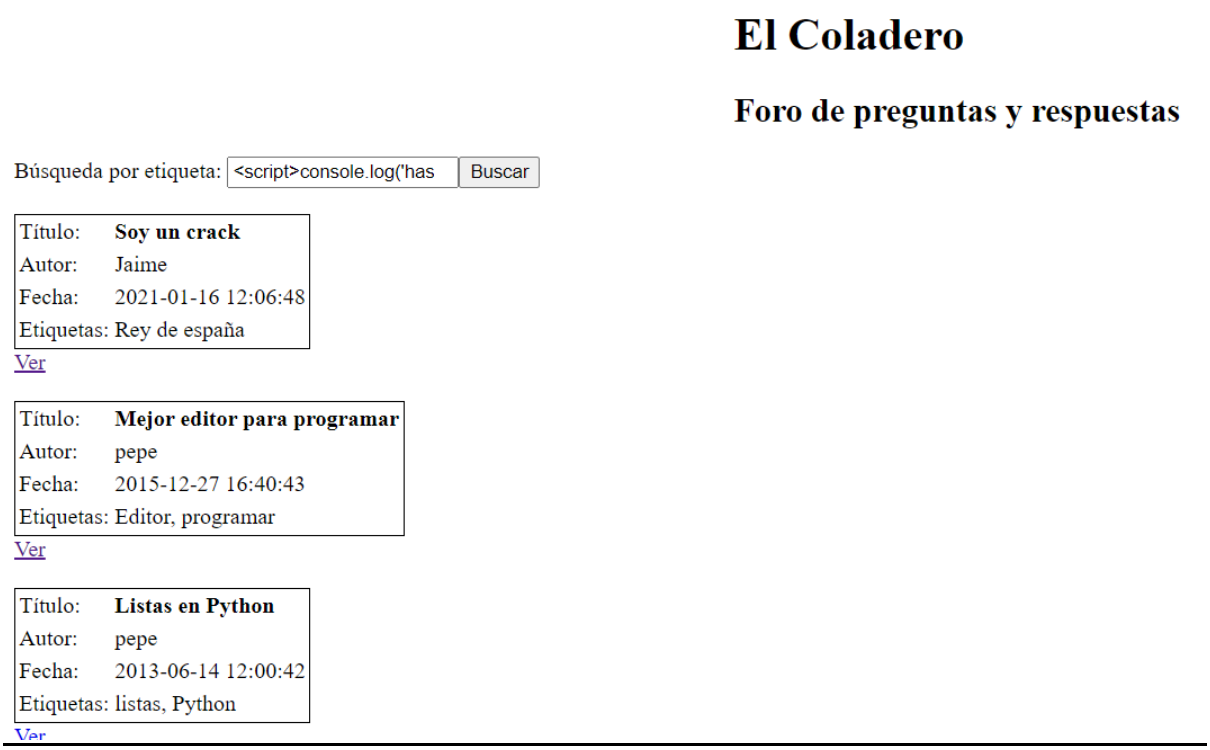
Causante de la vulnerabilidad:
Introducción de un script HTML en una solicitud de búsqueda de preguntas por etiqueta para que la página devuelta ejecute el código que se le ha indicado

Situaciones peligrosas o no deseadas que puede provocar:

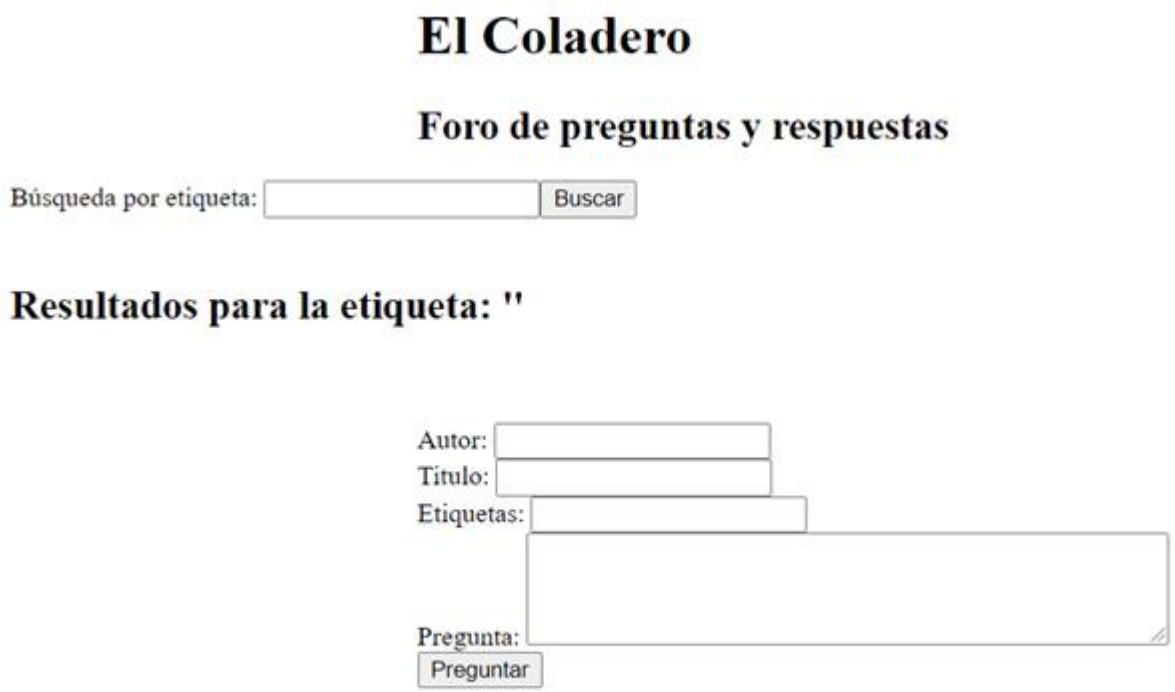
- **El script puede robarnos las cookies**
- **Obtener el ID de sesión**
- **Modificar el DOM de nuestra página web**

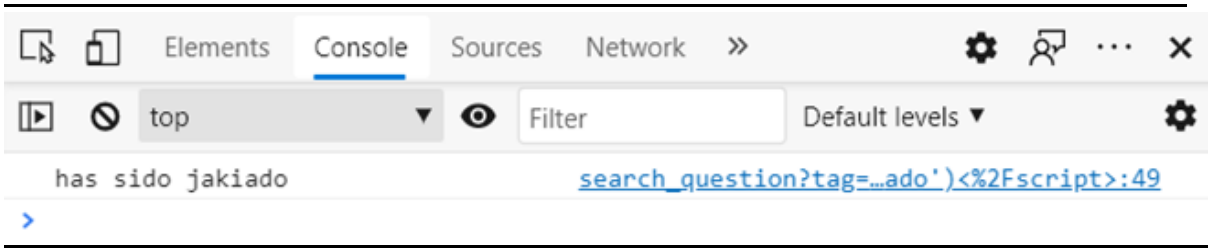
Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):

1) **Entramos en la página y, en el campo de buscar por etiqueta, introducimos un script html, en este caso *<script>console.log('has sido jakiado')</script>***



2) **Como podemos ver, ahora la página no devuelve al usuario ningún texto, sino que en su lugar ejecuta el script que le hemos introducido con anterioridad. Este script no se guarda en el servidor como en la vulnerabilidad anterior, sino que solo se le devuelve al usuario que hizo la búsqueda.**

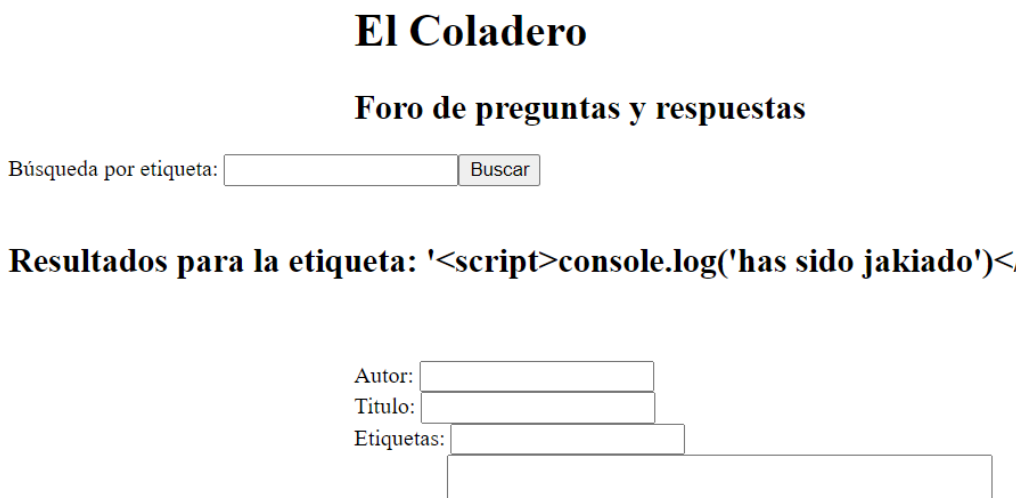




Medidas para mitigar la vulnerabilidad

Podemos hacer lo mismo que hicimos en el apartado anterior. Haciendo un replace de los corchetes por su correspondiente símbolo en código html, podemos hacer que la página lo interprete como una cadena de texto y no como un script a ejecutar. Para mitigar también otras vulnerabilidades se puede usar la función html.escape(), con carácter más general.

```
@app.route('/search_question', methods=['GET'])
def search_question():
    tag = request.args['tag']
    tag = tag.replace("<", "&#60")
    tag = tag.replace(">", "&#62")
    conn = sqlite3.connect(DBPATH)
    cur = conn.cursor()
    qbody = """SELECT id,author,title,time,tags
              FROM Questions
              WHERE tags LIKE :pattern
```



Vulnerabilidad 3

Ruta(s) de la aplicación involucrada(s)

127.0.0.1:5000/insert_question

Tipo de Vulnerabilidad

Inserción SQL

Causante de la vulnerabilidad:

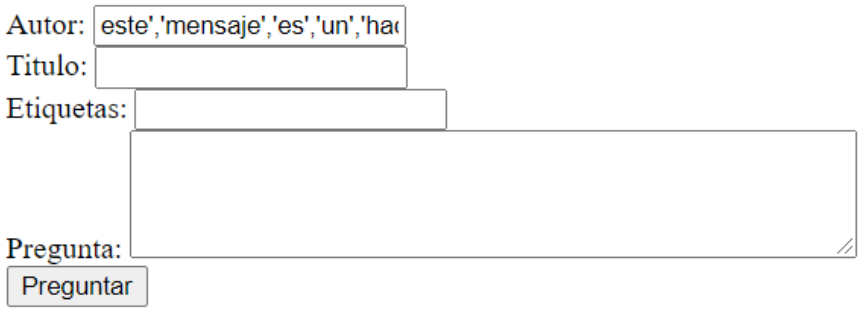
Introducción de código SQL en una pregunta del foro para manipular la base de datos interna.

Situaciones peligrosas o no deseadas que puede provocar:

- **Ejecución de instrucciones que podrían llegar a eliminar información de la base de datos.**
- **Obtención de todos (o parte de) los datos almacenados en la base de datos**
- **Obtención de las definiciones de la base de datos**

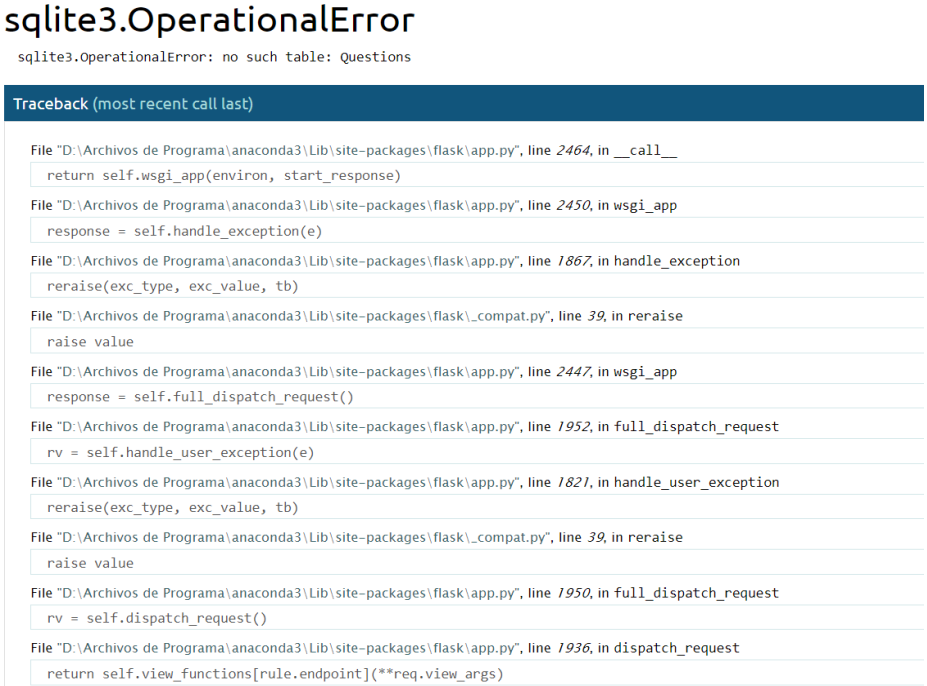
Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):

1) **Nos dirigimos al apartado de introducir una nueva pregunta, y, por ejemplo, en el apartado de autor introducimos el siguiente código:**
   este','mensaje','es','un','hackeo'); DROP TABLE Questions; --



2) **Pulsamos Preguntar y añadimos la pregunta**

3) **Al hacer click en volver, salta el siguiente error puesto que la tabla de preguntas ha sido borrada por la secuencia introducida.**



Medidas para mitigar la vulnerabilidad:

Si utilizamos el método html.escape() antes de insertar los parámetros en la base de datos, nos podemos asegurar de que aquello que entre en ella no contendrá caracteres que puedan interferir en su funcionamiento (se sustituirán por sus equivalentes en código html).

```
@app.route('/insert_question', methods=['POST'])
def insert_question():
    author = request.form['author']
    author = html.escape(author)
    title = request.form['title']
    title = html.escape(title)
    tags = request.form['tags']
    tags= html.escape(tags)
    body = request.form['body']
    body = html.escape(body)

    conn = sqlite3.connect(DBPATH)
    cur = conn.cursor()
    qbody = """INSERT INTO Questions(author, title, tags, body, time)
              VALUES ('{0}','{1}','{2}','{3}',CURRENT_TIMESTAMP)"""
    query = qbody.format(author, title, tags, body)
    cur.executescript(query)
```

De esta forma, al insertar lo mismo que hicimos antes, ahora el resultado sería este otro:

El Coladero

Foro de preguntas y respuestas

Búsqueda por etiqueta: Buscar

Título:
Autor: este','mensaje','es','un','hackeo'); DROP TABLE Questions; --
Fecha: 2021-01-16 13:33:57
Etiquetas:
Ver

Más info