

# GIW 2022-23

# Práctica 09

# Grupo 02

# Autores: Diego Revenga González, Raúl Blas Ruiz, Jorge Bello Martin, Eva Lucas Leiro

## VULNERABILIDAD: Inyección SQL

Ruta de la aplicación involucrada

/insert\_question

Tipo de vulnerabilidad

Inyección SQL

Causante de la vulnerabilidad

La query se ejecuta con `.executescript()`, que permite hacer varias consultas SQL. Si se hubiera utilizado `.execute()`, solo se habría ejecutado la primera consulta, lo que habría evitado el DROP TABLE.

Además, la query se crea con un `.format` de python, lo que permite hacer varias queries dentro de esta.

Situaciones peligrosas o no deseadas que puede provocar

Se podría terminar la query con un `“;”` y justo después hacer un drop table.

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

1º. Publicar la siguiente pregunta:

Autor:

Título:

Etiquetas:

Pregunta:

2º. Hecho. Para comprobar que la tabla Questions se ha borrado, se puede ir a la raíz de la aplicación y ver la excepción.

```
OperationalError
sqlite3.OperationalError: no such table: Questions

Traceback (most recent call last)
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 2548, in __call__
    return self.wsgi_app(environ, start_response)
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 2528, in wsgi_app
    response = self.handle_exception(e)
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 2525, in wsgi_app
    response = self.full_dispatch_request()
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 1822, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 1820, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\Users\reven\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\app.py", line 1796, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "C:\Users\reven\Desktop\GIW\Practica1\coladero.py", line 49, in show_all_questions
    cur.execute(query)

sqlite3.OperationalError: no such table: Questions

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:
```

Medidas para mitigar la vulnerabilidad

Utilizar el método `cur.execute(query, params)` propio de `sqlite3`, que evita que los parámetros sean código SQL y que se ejecuten varias consultas SQL.

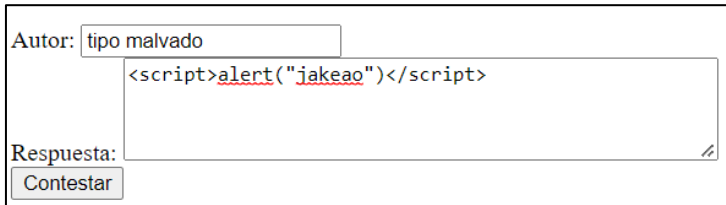
Alternativamente se podría comprobar manualmente que los parámetros no contengan código SQL.

# GIW 2022-23

# Práctica 09

# Grupo 02

# Autores: Diego Revenga González, Raúl Blas Ruiz, Jorge Bello Martin, Eva Lucas Leiro

VULNERABILIDAD: XSS Persistente	
Ruta de la aplicación involucrada	
/insert_reply /show_question	
Tipo de vulnerabilidad	
XSS Persistente	
Causante de la vulnerabilidad	
En /insert_reply no se comprueba la respuesta del usuario. Esta puede contener etiquetas HTML como <script> y eso se guardaría tal cual en la base de datos. En /show_question no se comprueba la respuesta de la consulta a la base de datos y directamente se mete en la template HTML que se da como salida al usuario.	
Situaciones peligrosas o no deseadas que puede provocar	
Un atacante podría hacer una respuesta en la que incluye código HTML y cuando cualquier otro usuario vaya a ver las respuestas y se cargue la maliciosa, se cargará también el código HTML malicioso.	
Ejemplo paso a paso de cómo explotar la vulnerabilidad (con capturas de pantalla)	
1º. Hacer una respuesta como la siguiente: 	
2º. Esperar a que otro usuario vaya a la pagina donde se ha posteado la respuesta y el script se ejecutará en su máquina.	
Medidas para mitigar la vulnerabilidad	
Verificar todos los parámetros que incluye un usuario en una solicitud. En este caso, verificar que ni el campo autor ni la respuesta incluyen código HTML. Hacer esto antes de insertar cualquier valor en la base de datos. Y para mayor seguridad hacerlo también después de cada consulta a la base de datos cuya respuesta deba ser sacada en una template HTML.	

# GIW 2022-23

# Práctica 09

# Grupo 02

# Autores: Diego Revenga González, Raúl Blas Ruiz, Jorge Bello Martin, Eva Lucas Leiro

VULNERABILIDAD: XSS Reflejado
Ruta de la aplicación involucrada
/search_question
Tipo de vulnerabilidad
XSS Reflejado
Causante de la vulnerabilidad
La etiqueta introducida por el usuario no se valida, escapando las etiquetas html y al meterla directamente en la template de la web devuelta, se pueden introducir scripts en ella.
Situaciones peligrosas o no deseadas que puede provocar
Un atacante le hace llegar un link a su víctima con un parámetro que contiene una etiqueta <script> con código JavaScript malicioso en ella.
Ejemplo paso a paso de cómo explotar la vulnerabilidad (con capturas de pantalla)
<p>1º. El atacante crea el siguiente enlace:</p> <p><a href="http://localhost:5000/search_question?tag=%3Cscript%3Ealert(%22hola%22)%3C%2Fscript%3E">http://localhost:5000/search_question?tag=%3Cscript%3Ealert(%22hola%22)%3C%2Fscript%3E</a></p> <p>(Este enlace es fácilmente creable a partir de la búsqueda por etiqueta de la página raíz.)</p> <div><div>Búsqueda por etiqueta:</div><div><input type="text" value='&lt;script&gt;alert("hola")&lt;/scri'/></div><div>Buscar</div></div>
<p>2º. El atacante le hace llegar a su víctima el enlace y cuando esta lo abra, se ejecutará el script. (En este caso es un simple alert, pero podría ser cualquier cosa).</p>
Medidas para mitigar la vulnerabilidad
Validar TODOS los parámetros que lleguen en cualquier solicitud de un usuario, escapando secuencias HTML.