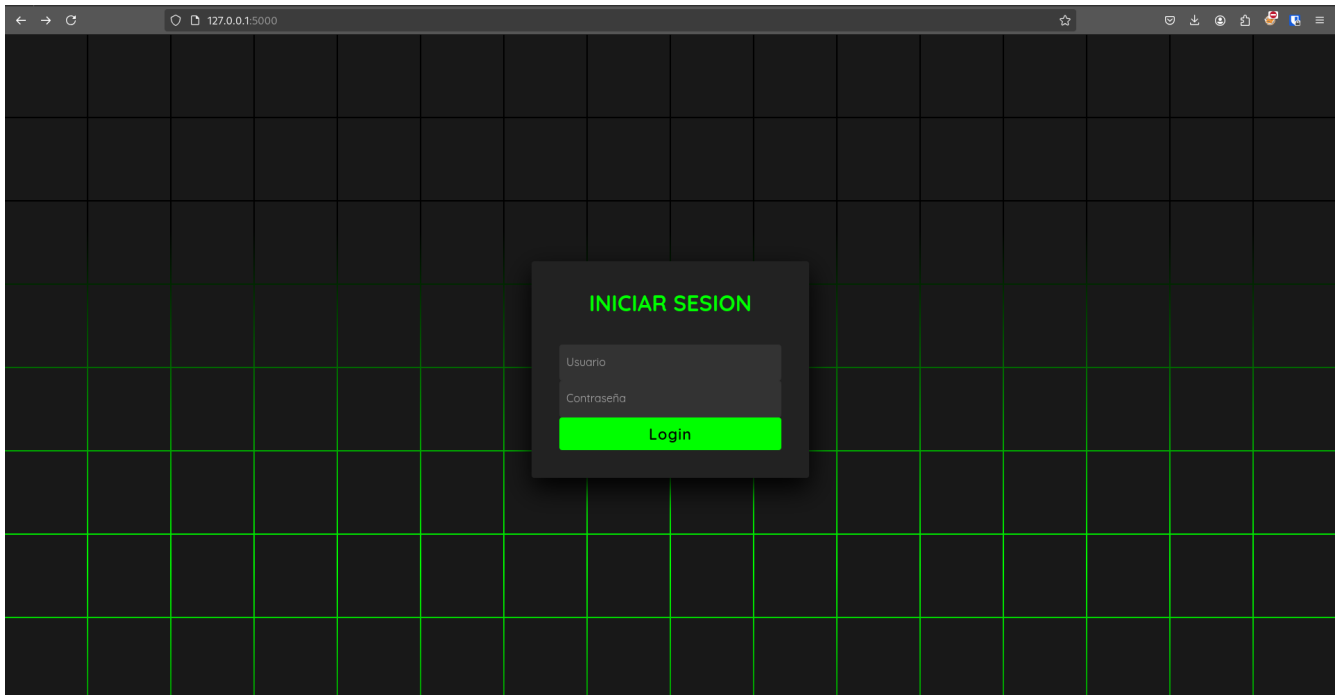


ShadowPass Writeup

Para empezar, en la descripción del reto nos dan información que nos sugiere y nos da a entender como se resuelve este reto, especialmente la siguiente línea

“La fuerza y la violencia nunca son la solución, ¡Hey, mira, ¿qué es ese error?!”

Al acceder a la dirección IP y puerto mediante un navegador web, a primera vista tenemos un formulario de inicio de sesión



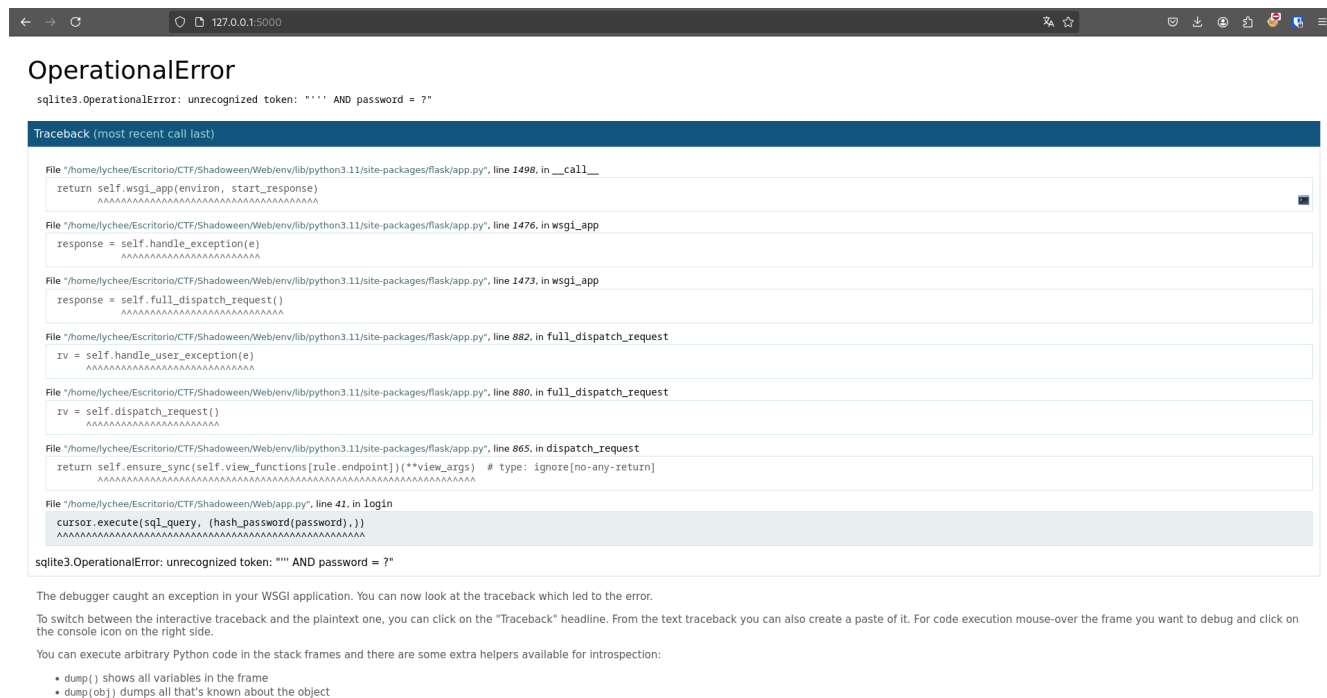
Retomando la información que nos dan del reto, a primera vista, se nos indica que en algún lugar, puede haber un error, y de una manera mas sutil, podemos entender que la fuerza bruta no es necesario para poder completar el reto

Primero podemos intentar con credenciales por defecto como **root:password** o **admin:admin**



Con esto podemos identificar que se puede realizar una enumeración de usuarios, lo que sería la primera vulnerabilidad identificada, ahora necesitamos encontrar el error que se menciona en la descripción

Un de las vulnerabilidades mas comunes en los formularios, especialmente en los de inicio de sesión, es la **inyección SQ**, la cual nos permite inyectar instrucciones SQL en el servidor, ahora procedemos a verificar si podemos romper la instrucción utilizando comillas dobles o simples en los 2 campos



```
OperationalError
sqlite3.OperationalError: unrecognized token: '''' AND password = ?"

Traceback (most recent call last)
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 1498, in __call__
    return self.wsgi_app(environ, start_response)
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 1476, in wsgi_app
    response = self.handle_exception(e)
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 1473, in wsgi_app
    response = self.full_dispatch_request()
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 882, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 880, in full_dispatch_request
    rv = self.dispatch_request()
File "/home/lychee/Escritorio/CTF/Shadoween/Web/env/lib/python3.11/site-packages/flask/app.py", line 865, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  # type: ignore[no-any-return]
File "/home/lychee/Escritorio/CTF/Shadoween/Web/app.py", line 41, in login
    cursor.execute(sql_query, (hash_password(password),))
sqlite3.OperationalError: unrecognized token: '''' AND password = ?"

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:

• dump() shows all variables in the frame
• dump(obj) dumps all that's known about the object
```

Con este error podemos confirmar que el formulario es vulnerable a inyecciones SQL y si prestamos atención al mensaje, podemos darnos cuenta de que el parámetro vulnerable es **username**

Con esta información, podemos inyectar una sentencia SQL que nos permita saltarnos el inicio de sesión, para construir esta sentencia, tenemos que entener que es lo que se esta ejecutando

SELECT * FROM users WHERE username = '{username}' AND password = ?

Esta es la sentencia completa que se usa para validar el inicio de sesión, lo que hace es acceder a la tabla **users** y buscar el usuario que coincida con el nombre de usuario y la contraseña que se pone en el inicio de sesión, si ambos datos coinciden, se valida la sesión Sabiendo esto, la instrucción que tenemos que inyectar debe seleccionar un usuario valido y se debe buscar la manera de evitar que se compare la contraseña, ya que no sabemos cual es

Como sabemos que admin es un usuario valido, el payload debe llevar ese usuario, ahora para la contraseña, se puede inyectar una nueva sentencia que cumpla con una condición, sabiendo esto, podemos construir el siguiente payload:

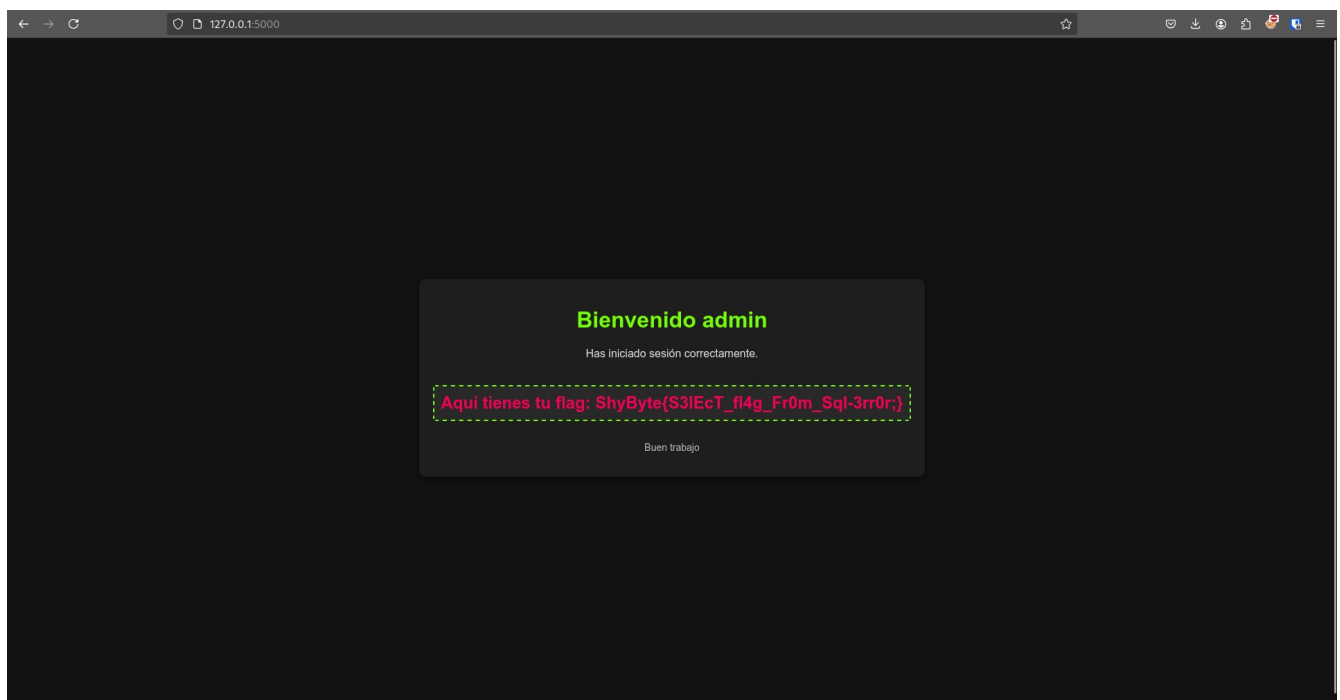
admin' OR '1'='1

La explicación de este payload es simple: primero tenemos el usuario válido, luego rompemos la sentencia SQL con una comilla simple y después inyectamos una condición que siempre va a ser verdadera, ya que 1 es igual a 1 y dejamos sin cerrar la última comilla, ya que el código de la aplicación, ya le asigna esa comilla, por lo que ponerle otra nos daría un error, para entenderlo mejor, esta es la sentencia completa que se ejecuta:

```
SELECT * FROM users WHERE username = 'admin' OR '1'='1' AND password = ?
```

En SQL el operador OR tiene una posición jerárquica mayor que el operador AND, por lo que al final de la consulta, se va a buscar que el usuario sea igual a admin y después se va a evaluar la condición OR '1'='1' la cual siempre es verdadera, por último se va a evaluar que la contraseña de la base de datos sea igual a la que proporcionamos en el inicio de sesión, pero esta condición se vuelve irrelevante ya que la condición anterior siempre será verdadera, lo que hace que la consulta devuelva datos

Una vez entendido esto, podemos usar el payload **admin' OR '1'='1** para evitar la condición de la contraseña y poder saltar el inicio de sesión



Y de esta manera hacemos un bypass del inicio de sesión sin haber obtenido la contraseña en ningún momento