NMAP2DB - NMAP scans management

Version-1.0.0

Author: Rafael Martinez Guerrero (University of Oslo)
E-mail: rafael@postgresql.org.es
Source: https://github.com/rafaelma/nmap2db

Contents

Introduction **1**

Main features **2**

Architecture and components **2**

Installation **2**

System requirements 2

Installing from source 3

Installing via RPM packages 3

Installing via Deb packages 3

Installing the nmap2db database 3

Configuration **4**

Scan servers 4

System administration and maintenance **4**

NMAP2DB shell **5**

clear 5

quit 5

shell 6

show_history 6

Submitting a bug **7**

Authors **7**

License and Contributions **7**

Introduction

NMAP2DB is a tool for managing nmap scans and save the result in a PostgreSQL database.

It is designed to run thousands of scans a day , save the results in a database and use the nmap2db shell to interact with the system.

The NMAP2DB code is distributed under the GNU General Public License 3 and it is written in Python and PL/PgSQL. It has been developed and tested by members of the CERT group at the Center for Information Technology at the University of Oslo.

Main features

The main features of NMAP2DB are:

- Central database with metadata and raw information.
- NMAP2DB shell for interaction with the system.
- Management of multiple networks
- Management of multiple scan types
- Scans scheduling
- Written in Python and PL/PgSQL
- Distributed under the GNU General Public License 3

Architecture and components

The components forming part of Nmap2db could be listed as follows:

- **Scan servers:** One or several servers running NMAP2DB. They will use nmap to execute the scans defined in the system and will access via `libpq` the nmap2db database to save and retrieve the data.
- **nmap2db DB**: Central postgreSQL database used by NMAP2DB. All scan servers need access to this database.
- **NMAP2DB shell:** This is a program that must be run in a text terminal. It can be run in any of the scan servers. It is the console used to manage NMAP2DB.

Installation

You will have to install the NMAP2DB software in all the servers that are going to be used to run nmap scans.

System requirements

- Linux/Unix
- Python 2.6 or 2.7
- Python modules:
  - psycopg2
  - argparse
- PostgreSQL >= 9.2 for the `nmap2db` database
- NMAP >= xxxx

Before you install NMAP2DB you have to install the software needed by this tool

In systems using `yum`, e.g. Centos, RHEL, ...:

```
yum install python-psycopg2 python-argparse nmap
```

In system using `apt-get`, e.g. Debian, Ubuntu, ...:

```
apt-get install python-psycopg2 python-argparse nmap
```

If you are going to install from source, you need to install also these packages: `python-dev(el), python-setuptools, git, make, rst2pdf`

In systems using `yum`:

```
yum install python-devel python-setuptools git make rst2pdf
```

In system using `apt-get`:

```
apt-get install python-dev python-setuptools git make rst2pdf
```

Installing from source

The easiest way to install nmap2db from source is to get the last version from the master branch at the GitHub repository.

```
[root@server]# cd
[root@server]# git clone https://github.com/rafaelma/nmap2db.git

[root@server]# cd nmap2db
[root@server]# ./setup2.py install
.....
```

This will install all users, groups, programs, configuration files, logfiles and the nmap2db module in your system.

Installing via RPM packages

RPM packages for CentOS 6 and RHEL6 are available at https://github.com/rafaelma/nmap2db/releases

Install the RPM package with:

```
[root@server]# rpm -Uvh nmap2db-<version>.rpm
```

Installing via Deb packages

Deb packages for Debian7 are available at https://github.com/rafaelma/nmap2db/releases

Install the Deb package with:

```
[root@server]# dpkg -i nmap2db_<version>.deb
```

Installing the nmap2db database

After the requirements and the NMAP2DB software are installed, you have to install the `nmap2db` database in a server running PostgreSQL. This database is the core of the NMAP2DB tool and it is used to save all the metadata needed to manage the system.

You can get this database from the directory `sql/` in the source code or under the directory `/usr/share/nmap2db` if you have installed NNAMP2DB via `source`, `rpm` or `deb` packages.

```
psql -h <dbhost.domain> -f /usr/share/nmap2db/nmap2db.sql
```

There is another file in this directory named `nmap2pg_table_partition.sql`. This file can be used to install and configure partitioning of the main tables used by NMAP2DB. We recommend to use table partitioning when using NMAP2DB. The nmap2db database can become very large if you have a large network and you want to keep some historic data and partitioning will help to have a good performance when searching for data in the database.

Run this command to install partitioning support.

```
psql -h <dbhost.domain> -f /usr/share/nmap2db/nmap2db_table_partition.sql
```

Configuration

Scan servers

A scan server needs to have access to the `nmap2db` database. This can be done like this:

1. Update `/etc/nmap2db/nmap2db.conf` with the database parameters needed by NMAP2DB to access the central database. You need to define `host` or `hostaddr`, `port`, `dbname`, `database` under the section `[nmap2db_database]`.

   You can also define a `password` in this section but we discourage to do this and recommend to define a `.pgpass` file in the home directory of the users `root` and `nmap2db` with this information, e.g.:

   ```
   <dbhost.domain>:5432:nmap2db:nmap2db_role_rw:PASSWORD
   ```

   and set the privileges of this file with `chmod 400 ~/.pgpass`.

   An even better solution will be to use `cert` autentication for the nmap2db database user, so we do not need to save passwords values.

2. Update and reload the `pg_hba.conf` file in the postgreSQL server running the `nmap2db` database, with a line that gives access to the nmap2db database from the new backup server. We recommend to use a SSL connection to encrypt all the traffic between the database server and the backup server, e.g.:

   ```
   hostssl   nmap2db   nmap2db_role_rw    <scan_server_IP>/32     md5
   ```

System administration and maintenance

If NMAP2DB is using table partitioning we have to run a job every month to maintain all the tables, triggers and indexes we use for this.

This job can be executed via cron everty month. Create this file `/etc/crond.d/nmap2db` with this content.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=your@email_address

01 00 01 * * root /usr/bin/psql -h <your.dbhost> -U nmap2db_role_rw nmap2db -c "SELECT c
```

The script `/etc/init.d/nmap2db_ctrl.sh` can be used to start or stop new nmap2db scan processes. This is a simple bash script that does not follow or implement any System V requirements and can not be used to start/stop nmap2db automatically when the server running NMAP2DB boots or shutdowns.

To start e.g. 40 nmap2db scan processes:

```
/etc/init.d/nmap2db_ctrl.sh -n 20 -c start
```

To stop all nmap2db scan processed:

```
::
```

/etc/init.d/nmap2db_ctrl.sh -c stop

NMAP2DB shell

The NMAP2DB interactive shell can be started by running the program `/usr/bin/nmap2db`

```
[nmap2db@scan_server]# nmap2db
Needs output
```

NOTE: It is possible to use the NMAP2DB shell in a non-interactive modus by running `/usr/bin/nmap2db` with a command as a parameter in the OS shell. This can be used to run NMAP2DB commands from shell scripts.e.g.:

```
Needs example
```

clear

This command clears the screen and shows the welcome banner

```
clear
```

This command can be run only without parameters. e.g.:

```
[nmap2db]$ clear

#############################################################
Welcome to the PostgreSQL Backup Manager shell (v.1.0.0)
#############################################################
Type help or \? to list commands.

[nmap2db]$
```

quit

This command quits/terminates the Nmap2db shell.

```
quit
```

A shortcut to this command is `\q`.

This command can be run only without parameters. e.g.:

```
[nmap2db]$ quit
Done, thank you for using Nmap2db
```

```
[nmap2db]$ \q
Done, thank you for using Nmap2db
```

shell

This command runs a command in the operative system.

```
shell [command]
```

Parameters:

- **[command]:** Any command that can be run in the operative system.

It exists a shortcut `[!]` for this command that can be used insteed of `shell`. This command can be run only with parameters. e.g.:

```
[nmap2db]$ ! ls -l
total 88
-rw-rw-r--. 1 vagrant vagrant   135 May 30 10:04 AUTHORS
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 bin
drwxrwxr-x. 4 vagrant vagrant  4096 May 30 10:03 docs
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 etc
-rw-rw-r--. 1 vagrant vagrant     0 May 30 10:04 INSTALL
-rw-rw-r--. 1 vagrant vagrant 35121 May 30 10:04 LICENSE
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 pgbackman
-rw-rw-r--. 1 vagrant vagrant   797 May 30 10:04 README.md
-rwxrwxr-x. 1 vagrant vagrant  4087 May 30 10:04 setup.py
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 sql
drwxrwxr-x. 4 vagrant vagrant  4096 May 30 10:03 vagrant
```

show_history

Show the list of commands that have been entered during the Nmap2db shell session.

```
show_history
```

A shortcut to this command is `\s`. One can also use the *Emacs Line-Edit Mode Command History Searching* to get previous commands containing a string. Hit `[CTRL]+[r]` in the Nmap2db shell followed by the search string you are trying to find in the history.

This command can be run only without parameters. e.g.:

```
[nmap2db]$ show_history

[0]: help
[1]: help support
[2]: help show_history
[3]: shell df -h | grep /srv/pgbackman
[4]: show_history
[5]: help
[6]: show_history
```

Submitting a bug

NMAP2DB has been extensively tested, and is currently being used in production. However, as any software, NMAP2DB is not bug free.

If you discover a bug, please file a bug through the GitHub Issue page for the project at: https://github.com/rafaelma/nmap2db/issues

Authors

In alphabetical order:

Rafael Martinez Guerrero
E-mail: rafael@postgresql.org.es / rafael@usit.uio.no
PostgreSQL-es / University Center for Information Technology (USIT), University of Oslo, Norway

License and Contributions

NMAP2DB is the property of Rafael Martinez Guerrero / PostgreSQL-es and USIT-University of Oslo, and its code is distributed under GNU General Public License 3.