PaSSHport Documentation

LibrIT

Mar 11, 2022

Contents

Your main adminsys leaves your company. Are you sure all his ssh access are revoked? What about the interns? The consultants?... Let's fix this.

CHAPTER 1

Table of content :

1.1 Introduction to PaSSHport

1.1.1 What is PaSSHport ?

PaSSHport is a software that allows you to control the SSH access of your IT components : Linux/Unix servers, network switchs, routers, WiFi access points, and any appliances that is accessed by SSH. In three words : who accesses what ?

PaSSHport has been written with the following in mind :

- Similar to SSHgate
- Two main objects : targets and users (we'll see below what are those)
- Objects can be grouped : targetgroups and usergroups
- Record all sessions of users
- Can be fully configure and used from the command line interface
- Can do Secured Copy (scp)
- REST API based communication between components so that it can be easily integrated in an automated IT environment
- Use only OpenSource technologies

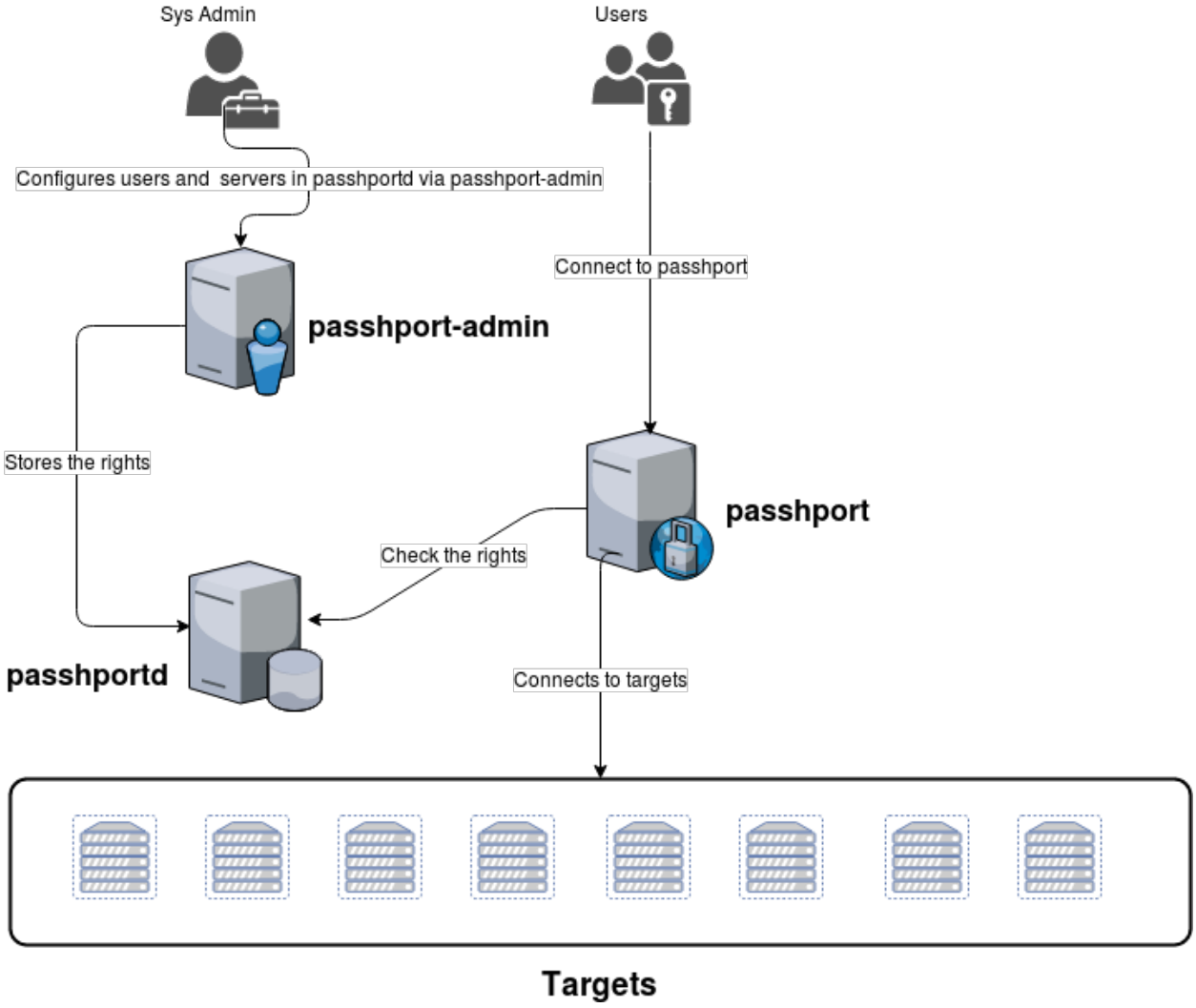
Please read below to understand the main components, and how they work together.

1.1.2 Components

PaSSHport project is composed with 3 main programs :

- passhportd : the daemon that verify access rights, and store configuration

- passhport : the script that receive the connection (it does NOT replace the SSH server). Think of it has the shell a user falls into when connecting to a PaSSHport gateway
- passhport-admin : the script that is used to configure passhportd. SysAdmins will use it to add a *user*, a *target*, a *usergroup*, a *targetgroup*, and combine those to configure accesses



Now let's go to the installation process...

1.2 Installation and configuration

This chapter describes the installation on different distro, and the main configuration principles.

1.2.1 Installation on Debian 8, 9 or 10

The followings shows you how to install and run PaSSHport on Debian 8 (Jessie), 9 (Stretch) or 10 (Buster). We start from a minimal installation of Debian (available here), **with openssh-server and curl** packages installed.

The easy, automated way

You can review the installation script here.

You can run it directly from command line (please ensure that curl is installed : `apt install curl`):

```
root@debian:~# bash <(curl -s https://raw.githubusercontent.com/librit/passhport/
↪master/tools/passhport-install-script-debian.sh)
```

Once finished, you can go to the Getting Started chapter.

The long, manual way

To understand what you do on your system when you install PaSSHport, follow the instructions below, that are more or less the step by step commands from the automated installation script.

First of all, we'll need to update your repositories :

```
root@debian:~# apt update
```

We will install python3-pip, and other packages that we'll need later for this tutorial (it will get ~+100MB from the archives, so be patient) :

```
root@debian:~# apt install python3-pip git openssl virtualenv libpython3-dev
```

Next we will need to add a system user called « passhport », and switch to it :

```
root@debian:~# useradd --home-dir /home/passhport --shell /bin/bash --create-home␣
↪passhport
root@debian:~# su - passhport
passhport@debian:~$
```

Let's get passhport sources from github :

```
passhport@debian:~$ git clone http://github.com/LibrIT/passhport.git
Clonage dans 'passhport'...
remote: Counting objects: 2713, done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 2713 (delta 19), reused 0 (delta 0), pack-reused 2661
Réception d'objets: 100% (2713/2713), 482.76 KiB | 396.00 KiB/s, fait.
Résolution des deltas: 100% (1633/1633), fait.
passhport@debian:~$
```

We now need to create a virtual-env for passhport user :

```
passhport@debian:~$ virtualenv -p python3 passhport-run-env
```

Now that we have our virtual-env, we install the python's modules we'll need for PaSSHport :

```
passhport@debian:~$ /home/passhport/passhport-run-env/bin/pip install -r /home/
↪passhport/passhport/requirements.txt
```

Now, let's start the real thing…

PaSSHport will need to write some logs, so, as root, we'll create a directory in « /var/log », and give the ownership to the « passhport » user:

```
root@debian:~# mkdir -p /var/log/passhport/
root@debian:~# chown passhport:passhport /var/log/passhport/
```

We'll also create the config directory, and copy the differents config file :

```
root@debian:~# mkdir /etc/passhport
root@debian:~# cp /home/passhport/passhport/passhport/passhport.ini /etc/passhport/.
root@debian:~# cp /home/passhport/passhport/passhport-admin/passhport-admin.ini /etc/
↪passhport/.
root@debian:~# cp /home/passhport/passhport/passhportd/passhportd.ini /etc/passhport/.
```

We'll also need to make some modifications in those config file, if you run passhportd on a distant server. Here we'll change the default listening address (localhost) to the real IP of our server.

First, passhportd :

```
root@debian:~# vim /etc/passhport/passhportd.ini
```

Change the « LISTENING_IP » parameter, to the IP address of your server :

```
# Passhportd configuration file. You should copy it to
# /etc/passhport/passhportd.ini if you want to do modifications
[SSL]
SSL = True
SSL_CERTIFICAT = /home/passhport/certs/cert.pem
SSL_KEY = /home/passhport/certs/key.pem

[Network]
LISTENING_IP = 192.168.122.56
PORT = 5000

[Database]
SQLALCHEMY_TRACK_MODIFICATIONS = True
SQLALCHEMY_DATABASE_DIR = /var/lib/passhport/
SQLALCHEMY_MIGRATE_REPO = /var/lib/passhport/db_repository
# For SQLite
SQLALCHEMY_DATABASE_URI = sqlite:////var/lib/passhport/app.db

[Environment]
# SSH Keyfile path
SSH_KEY_FILE = /home/passhport/.ssh/authorized_keys
# Python and passhport paths
PASSHPORT_PATH = /home/passhport/passhport/passhport/passhport
PYTHON_PATH = /home/passhport/passhport-run-env/bin/python3
```

Change the following parameter in /etc/passhport/passhport.ini and /etc/passhport/passhport-admin.ini :

```
PASSHPORTD_HOSTNAME = 192.168.122.56
```

We'll need ssh publickey, so we generate a 4096 bits RSA key (keys lengh can be longer):

```
root@debian:~# su - passhport
passhport@debian:~$ ssh-keygen -t rsa -b 4096 -N "" -f "/home/passhport/.ssh/id_rsa"
Generating public/private rsa key pair.
Your identification has been saved in /home/passhport/.ssh/id_rsa.
Your public key has been saved in /home/passhport/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0o6jkepqr2Phz0AKmLGRZh6PeVexP2gf5CGNPd+ksQ passhport@debian
```

(continues on next page)

(continued from previous page)

```
The key's randomart image is:
+---[RSA 4096]----+
| .    ....       |
|oo . o .+ +      |
|* + o ...= *     |
|.O   o oo + E    |
|=.    LibrIT .   |
|+.   .Rocks = .  |
|o.. o o .  . o   |
| =o. o .         |
|++B+.            |
+----[SHA256]-----+
passhport@debian:~$
```

This will be the key that'll be use by PaSSHport to connect to your hosts. You can also generate a ECDSA key if you wish :

```
passhport@debian:~$ ssh-keygen -t ecdsa -b 521 -N "" -f "/home/passhport/.ssh/id_ecdsa
↪"
```

Again as root, let's make the directory that'll contains the database (because we use SQLite for this tutorial) :

```
root@debian:~# mkdir -p /var/lib/passhport
root@debian:~# chown -R passhport:passhport /var/lib/passhport/
```

... then we'll have to change 3 paramaters in the passhportd config file (as root, edit «/etc/passhport/passhportd.ini») :

```
SQLALCHEMY_DATABASE_DIR         = /var/lib/passhport/
SQLALCHEMY_MIGRATE_REPO         = /var/lib/passhport/db_repository
SQLALCHEMY_DATABASE_URI         = sqlite:////var/lib/passhport/app.db
```

Now we can create the database and check that it has correcly been created:

```
root@debian:~# su - passhport
passhport@debian:~$ /home/passhport/passhport-run-env/bin/python /home/passhport/
↪passhport/passhportd/db_create.py
passhport@debian:~$ ls -la /var/lib/passhport/
total 172
drwxr-xr-x  3 passhport passhport   4096 févr. 28 16:10 .
drwxr-xr-x 25 root      root        4096 févr. 28 15:37 ..
-rw-r--r--  1 passhport passhport 159744 févr. 28 16:10 app.db
drwxr-xr-x  4 passhport passhport   4096 févr. 28 16:10 db_repository
passhport@debian:~$
```

We'll now need to create the certificate to secure the API. First, create the directory in which will be key and the cert, and make the directory rwx for passport only :

```
passhport@debian:~$ mkdir /home/passhport/certs
passhport@debian:~$ chmod 700 /home/passhport/certs
```

Create the RSA key :

```
[passhport@centos-7 ~]$ openssl genrsa -out "/home/passhport/certs/key.pem" 4096
```

There is a conf file provided for OpenSSL, to generate a minimal correct SSL cert. The file is :

`/home/passhport/passhport/tools/openssl-for-passhportd.cnf`

Edit it, and add DNS name you'll use to reach the API. For the tutorial, we'll use two hostnames :

```
[req]
distinguished_name      = req_distinguished_name
req_extensions          = v3_req
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer

[v3_req]
subjectAltName          = @alternate_names
basicConstraints        = CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer

[req_distinguished_name]

[ alternate_names ]
DNS.1 = 127.0.0.1
DNS.2 = localhost
DNS.3 = passhport.librit.fr
DNS.4 = entry.passhport.org
```

Now, generate the certificate using this command (put on multiple lines, so you can copy/paste easily), but please adapt the subject line (-subj) :

```
openssl req -new -key "/home/passhport/certs/key.pem" \
-config "/home/passhport/passhport/tools/openssl-for-passhportd.cnf" \
-out "/home/passhport/certs/cert.pem" \
-subj "/C=FR/ST=Ile De France/L=Ivry sur Seine/O=LibrIT/OU=DSI/CN=passhport.librit.fr
↪" \
-x509 -days 365 -sha256 \
-extensions v3_req
```

Once executed, you'll have a cert file next to the key file :

```
passhport@debian:~$ ls -la /home/passhport/certs/
total 16
drwx------ 2 passhport passhport 4096 févr. 28 18:00 .
drwxr-xr-x 8 passhport passhport 4096 févr. 28 17:46 ..
-rw-r--r-- 1 passhport passhport 2171 févr. 28 18:00 cert.pem
-rw------- 1 passhport passhport 3243 févr. 28 16:11 key.pem
passhport@debian:~$
```

As root, create some symlink to the two main *binaries*, passhportd and passhport-admin, so you can access it without typing full path :

```
root@debian:~# ln -s /home/passhport/passhport/tools/passhportd.sh /usr/bin/passhportd
root@debian:~# ln -s /home/passhport/passhport/tools/passhport-admin.sh /usr/bin/
↪passhport-admin
```

We now create the systemd service, and enables *passhportd* on startup :

```
root@debian:~# cp /home/passhport/passhport/tools/passhportd.service /etc/systemd/
↪system/passhportd.service
root@debian:~# systemctl daemon-reload
root@debian:~# systemctl enable passhportd
```

And now, we're ready to go, just launch passhportd daemon :

```
root@debian:~# systemctl start passhportd
```

You can check that passhportd is running, by curling the IP you previously configured in */etc/passhport/passhportd.ini*, on port 5000 :

```
root@debian:~# curl -s --insecure https://192.168.122.56:5000
passhportd is running, gratz!
root@debian:~#
```

Bravo ! You successfully installed PaSSHport. You may now go to the Getting Started chapter.

1.2.2 Installation on CentOS 7

The followings shows you how to install and run PaSSHport on CentOS 7. We start from a minimal installation of CentOS (available here).

The easy, automated way

You can review the installation script here.

You can run it directly from command line :

```
root@centos7:~# bash <(curl -s https://raw.githubusercontent.com/librit/passhport/
↪master/tools/passhport-install-script-centos7.sh)
```

Once finished, you can go to the Getting Started chapter.

The long, manual way

To understand what you do on your system when you install PaSSHport, follow the instructions below, that are more or less the step by step commands from the automated installation script.

Install the EPEL repository :

```
yum install epel-release
```

Install python34-pip and other packages that we'll need later for this tutorial (it will get ~100MB from the archives, so be patient) :

```
root@centos7:~# yum install python34-pip git openssl python34-devel gcc libffi-devel
```

Let's update pip :

```
root@centos7:~# pip3 install -U pip
```

Now, install virtual-env and a mandatory lib using pip :

```
root@centos7:~# pip3 install virtualenv pathlib2
```

Next we will need to add a system user called « passhport », and switch to it :

```
root@centos7:~# useradd --home-dir /home/passhport --shell /bin/bash --create-home␣
↪passhport
root@centos7:~# su - passhport
passhport@centos7:~$
```

We now need to create a virtual-env for passhport user :

```
passhport@centos7:~$ virtualenv -p python3 passhport-run-env
```

Let's get passhport sources from github :

```
passhport@centos7:~$ git clone http://github.com/LibrIT/passhport.git
Clonage dans 'passhport'...
remote: Counting objects: 2713, done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 2713 (delta 19), reused 0 (delta 0), pack-reused 2661
Réception d'objets: 100% (2713/2713), 482.76 KiB | 396.00 KiB/s, fait.
Résolution des deltas: 100% (1633/1633), fait.
passhport@centos7:~$
```

Now that we have our virtual-env, we install the python's modules we'll need for PaSSHport :

```
passhport@centos7:~$ /home/passhport/passhport-run-env/bin/pip install -r /home/
↪passhport/passhport/requirements.txt
```

Now, let's start the real thing...

PaSSHport will need to write some logs, so, as root, we'll create a directory in « /var/log », and give the ownership to the « passhport » user:

```
root@centos7:~# mkdir -p /var/log/passhport/
root@centos7:~# chown passhport:passhport /var/log/passhport/
```

We'll also create the config directory, and copy the differents config file :

```
root@centos7:~# mkdir /etc/passhport
root@centos7:~# cp /home/passhport/passhport/passhport/passhport.ini /etc/passhport/.
root@centos7:~# cp /home/passhport/passhport/passhport-admin/passhport-admin.ini /etc/
↪passhport/.
root@centos7:~# cp /home/passhport/passhport/passhportd/passhportd.ini /etc/passhport/
↪.
```

We'll also need to make some modifications in those config file, if you run passhportd on a distant server. Here we'll change the default listening address (localhost) to the real IP of our server.

First, passhportd :

```
root@centos7:~# vim /etc/passhport/passhportd.ini
```

Change the « LISTENING_IP » parameter, to the IP address of your server :

```
# Passhportd configuration file. You should copy it to
# /etc/passhport/passhportd.ini if you want to do modifications
[SSL]
SSL = True
SSL_CERTIFICAT = /home/passhport/certs/cert.pem
SSL_KEY = /home/passhport/certs/key.pem
```

(continues on next page)

(continued from previous page)

```
[Network]
LISTENING_IP = 192.168.122.56
PORT = 5000

[Database]
SQLALCHEMY_TRACK_MODIFICATIONS = True
SQLALCHEMY_DATABASE_DIR = /var/lib/passhport/
SQLALCHEMY_MIGRATE_REPO = /var/lib/passhport/db_repository
# For SQLite
SQLALCHEMY_DATABASE_URI = sqlite:////var/lib/passhport/app.db

[Environment]
# SSH Keyfile path
SSH_KEY_FILE = /home/passhport/.ssh/authorized_keys
# Python and passhport paths
PASSHPORT_PATH = /home/passhport/passhport/passhport/passhport
PYTHON_PATH = /home/passhport/passhport-run-env/bin/python3
```

Change the following parameter in /etc/passhport/passhport.ini and /etc/passhport/passhport-admin.ini :

```
PASSHPORTD_HOSTNAME = 192.168.122.56
```

We'll need ssh publickey, so we generate an 4096 bits RSA key:

```
root@centos7:~# su - passhport
passhport@centos7:~$ ssh-keygen -t rsa -b 4096 -N "" -f "/home/passhport/.ssh/id_rsa"
Generating public/private rsa key pair.
Your identification has been saved in /home/passhport/.ssh/id_rsa.
Your public key has been saved in /home/passhport/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0o6jkepqr2Phz0AKmLGRZh6PeVexP2gf5CGNPd+ksQ passhport@centos7
The key's randomart image is:
+---[RSA 4096]----+
| .    ....       |
|oo . o .+ +      |
|* + o ...= *     |
|.O   o oo + E    |
|=.    LibrIT .   |
|+.   .Rocks = .  |
|o.. o o .  . o   |
| =o. o .         |
|++B+.            |
+----[SHA256]-----+
passhport@centos7:~$
```

This will be the key that'll be use by PaSSHport to connect to your hosts. You can also generate a ECDSA key if you wish :

```
passhport@centos7:~$ ssh-keygen -t ecdsa -b 521 -N "" -f "/home/passhport/.ssh/id_
↪ecdsa"
```

Again as root, let's make the directory that'll contains the database (because we use SQLite for this tutorial) :

```
root@centos7:~# mkdir -p /var/lib/passhport
root@centos7:~# chown -R passhport:passhport /var/lib/passhport/
```

... then we'll have to change 3 paramaters in the passhportd config file (as root, edit «/etc/passhport/passhportd.ini») :

```
SQLALCHEMY_DATABASE_DIR         = /var/lib/passhport/
SQLALCHEMY_MIGRATE_REPO         = /var/lib/passhport/db_repository
SQLALCHEMY_DATABASE_URI         = sqlite:////var/lib/passhport/app.db
```

Now we can create the database and check that it has correcly been created:

```
root@centos7:~# su - passhport
passhport@centos7:~$ /home/passhport/passhport-run-env/bin/python /home/passhport/
↪passhport/passhportd/db_create.py
passhport@centos7:~$ ls -la /var/lib/passhport/
total 172
drwxr-xr-x  3 passhport passhport   4096 févr. 28 16:10 .
drwxr-xr-x 25 root      root        4096 févr. 28 15:37 ..
-rw-r--r--  1 passhport passhport 159744 févr. 28 16:10 app.db
drwxr-xr-x  4 passhport passhport   4096 févr. 28 16:10 db_repository
passhport@centos7:~$
```

We'll now need to create the certificate to secure the API. First, create the directory in which will be key and the cert, and make the directory rwx for passport only :

```
passhport@centos7:~$ mkdir /home/passhport/certs
passhport@centos7:~$ chmod 700 /home/passhport/certs
```

Create the RSA key :

```
[passhport@centos-7 ~]$ openssl genrsa -out "/home/passhport/certs/key.pem" 4096
```

There is a conf file provided for OpenSSL, to generate a minimal correct SSL cert. The file is :

`/home/passhport/passhport/tools/openssl-for-passhportd.cnf`

Edit it, and add DNS name you'll use to reach the API. For the tutorial, we'll use two hostnames (localhost added) :

```
[req]
distinguished_name      = req_distinguished_name
req_extensions          = v3_req
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer

[v3_req]
subjectAltName          = @alternate_names
basicConstraints        = CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer

[req_distinguished_name]

[ alternate_names ]
DNS.1 = localhost
DNS.2 = passhport.librit.fr
DNS.3 = entry.passhport.org
```

Now, generate the certificate using this command (put on multiple lines, so you can copy/paste easily), but please adapt the subject line (-subj) :

```
openssl req -new -key "/home/passhport/certs/key.pem" \
-config "/home/passhport/passhport/tools/openssl-for-passhportd.cnf" \
-out "/home/passhport/certs/cert.pem" \
```

(continues on next page)

(continued from previous page)

```
-subj "/C=FR/ST=Ile De France/L=Ivry sur Seine/O=LibrIT/OU=DSI/CN=passhport.librit.fr
↪" \
-x509 -days 365 -sha256 \
-extensions v3_req
```

Once executed, you'll have a cert file next to the key file :

```
passhport@centos7:~$ ls -la /home/passhport/certs/
total 16
drwx------ 2 passhport passhport 4096 févr. 28 18:00 .
drwxr-xr-x 8 passhport passhport 4096 févr. 28 17:46 ..
-rw-r--r-- 1 passhport passhport 2171 févr. 28 18:00 cert.pem
-rw------- 1 passhport passhport 3243 févr. 28 16:11 key.pem
passhport@centos7:~$
```

As root, create some symlink to the two main *binaries*, passhportd and passhport-admin, so you can access it without typing full path :

```
root@centos7:~# ln -s /home/passhport/passhport/tools/passhportd.sh /usr/bin/
↪passhportd
root@centos7:~# ln -s /home/passhport/passhport/tools/passhport-admin.sh /usr/bin/
↪passhport-admin
```

We now create the systemd service, and enables *passhportd* on startup :

```
root@centos7:~# cp /home/passhport/passhport/tools/passhportd.service /etc/systemd/
↪system/passhportd.service
root@centos7:~# systemctl daemon-reload
root@centos7:~# systemctl enable passhportd
```

And now, we're ready to go, just launch passhportd daemon :

```
root@centos7:~# systemctl start passhportd
```

You can check that passhportd is running, by curling the IP you previously configured in */etc/passhport/passhportd.ini*, on port 5000 :

```
root@centos7:~# curl -s --insecure https://192.168.122.56:5000
passhportd is running, gratz!
root@centos7:~#
```

Bravo ! You successfully installed PaSSHport. You may now go to the Getting Started chapter.

1.2.3 Use PostgreSQL as database backend

Install psycopg2 and psycopg2-binary python modules

If you did not use the packaged version of passhport (deb/rpm), proceed as follow. If you used the package version, go directly below, to the *PostgreSQL configuration*.

Before installing python libs, be sure to have *pg_config* in your $PATH and some postgres libraries.

For Debian, install *postgresql*

```
# apt install postgresql
```

For CentOS, install *postgresql* :

```
# yum install postgresql
```

If you want to use PostgreSQL as the database backend you'll need to add two python modules : psycopg2 and psycopg2-binary.

As passhport user, install psycopg2 and psycopg2-binary:

```
$ /home/passhport/passhport-run-env/bin/pip install psycopg2 psycopg2-binary
```

PostgreSQL configuration

Create a passhport user in you postgreSQL server (may be different on your distro, this is just an example) :

```
# su - postgres
$ createuser -D -S -R passhport && createdb -O passhport "passhport"
```

Add a password to postgreSQL passhport user :

```
$ psql
psql (9.2.18)
Type "help" for help.

postgres=# ALTER USER "passhport" WITH PASSWORD 'MySUpErp45sw0rD';
ALTER ROLE
postgres=# \q
$
```

Passhportd configuration

Change the configuration of the *passhportd.ini* file (`/etc/passhport/passhportd.ini`). You need to change the `SQLALCHEMY_DATABASE_URI` parameter to :

```
SQLALCHEMY_DATABASE_URI         = postgresql://passhport:MySUpErp45sw0rD@localhost/
↪passhport
```

As passhport (system) user, initialize the database :

```
$ /home/passhport/passhport-run-env/bin/python /home/passhport/passhport/passhportd/
↪db_create.py
```

Then you can launch *passhportd* (kill it before if it stills running) :

```
$ /home/passhport/passhport-run-env/bin/python /home/passhport/passhport/passhportd/
↪passhportd
```

PaSSHport now use PostgreSQL backend.

1.2.4 Use MySQL as database backend

Install PyMySQL python module

If you did not use the packaged version of passhport (deb/rpm), proceed as follow. If you used the package version, go directly below, to the *MySQL configuration*.

If you want to use MySQL as the database backend you'll need to add a python module : PyMySQL.

As passhport user, install PyMySQL :

```
$ /home/passhport/passhport-run-env/bin/pip install PyMySQL
```

MySQL configuration

Create a *passhport* database in you MySQL server (may be different on your distro, this is just an example) :

```
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE passhport;
Query OK, 1 row affected (0.00 sec)
```

Then create a user that'll have all rights on the *passhport* database :

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON passhport.* TO passhport@localhost␣
↪IDENTIFIED BY 'iwetoh3oochieshaRei4';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> Bye

#
```

passhportd configuration

Change the configuration of the *passhportd.ini* file (`/etc/passhport/passhportd.ini`). You need to change the `SQLALCHEMY_DATABASE_URI` parameter to :

```
SQLALCHEMY_DATABASE_URI         = mysql+pymysql://
↪passhport:iwetoh3oochieshaRei4@localhost/passhport
```

As passhport (system) user, initialize the database :

```
$ /home/passhport/passhport-run-env/bin/python /home/passhport/passhport/passhportd/
↪db_create.py
```

Re-launch passhportd (as root) :

```
# systemctl restart passhportd
```

PaSSHport now use MySQL backend.

1.2.5 Add bash_completion capability to passhport-admin

We provide a bash_completion file, so that you can use [TAB] to auto-complete the passhport-admin command.

As root, copy the provide file to */etc/bash_completion/passhport-admin* directory, and source it :

```
# cp /home/passhport/passhport/tools/passhport-admin.bash_completion /etc/bash_
↪completion.d/passhport-admin
# . /etc/bash_completion.d/passhport-admin
```

You can now do things like these :

```
# passhport-admin [TAB][TAB]
target user targetgroup usergroup
# passhport-admin t[TAB]
# passhport-admin target[TAB]
target targetgroup
# passhport-admin targetg[TAB]
# passhport-admin targetgroup [TAB][TAB]
list search show create edit adduser rmuser
addtarget rmtarget addusergroup rmusergroup
addtargetgroup rmtargetgroup delete


# passhport-admin user show [TAB][TAB]
john rachel alfred bruce kim jared
# passhport-admin user show j[TAB]
john jared
# passhport-admin user show ja[TAB]
# passhport-admin user show jared
```

The end.

1.2.6 Add a WSGI server in front of PaSSHport

PaSSHport is based on Flask, the builtin web server can handle only one request at a time. It means that you can't really use it in production environnement...

In order to handle more requests, we duplicate the PaSSHport process for every connection. Apache provides such a solution with WSGI... And it's quite easy to activate it.

Installation

On Debian :

```
apt install apache2 libapache2-mod-wsgi-py3
```

Configuration

Create a new apache vhost file with this content:

```
Listen 5000
<VirtualHost *:5000>
    ServerName passhport
```

(continues on next page)

(continued from previous page)

```
    SSLEngine               on
    SSLCertificateFile      /home/passhport/certs/cert.pem
    SSLCertificatekeyFile   /home/passhport/certs/key.pem

    WSGIDaemonProcess passhport user=passhport group=passhport threads=5
    WSGIScriptAlias / /home/passhport/passhport/tools/passhportd.wsgi
    <Directory /home/passhport/ >
        WSGIProcessGroup passhport
        WSGIApplicationGroup %{GLOBAL}
        # passhportd don't provides authentication, please filter by IP
        Require ip 127.0.0.1/8 ::1/128
    </Directory>
</VirtualHost>
```

Activate

First kill the current passhport process

```
pkill passhportd
```

Deactivate default website and activate this one:

```
a2dissite 000-default
a2enmod ssl
a2ensite passhport.conf

systemctl restart apache2
```

and Voilà.

1.2.7 Renew passhportd TLS certificate

Explanation

If you installed PaSSHport a year from now, you may encounter this message when you try to connect :

```
# ssh passhport@passhport.example.com
No such user in PaSSHport database.
tip: it can be a SSL certificate misconfiguration.
Connection to passhport.example.com closed
#
```

This usually means that passhport (the script) can't connect to passhportd, and the most common cause is that the TLS certificate generated on installation is outdated.

```
passhport@passhport-srv:~$ openssl x509 -in /home/passhport/certs/cert.pem -noout -
↪text | grep Validity -A 2
      Validity
          Not Before: Sep 11 10:48:55 2020 GMT
          Not After : Sep 11 10:48:55 2021 GMT
passhport@passhport-srv:~$
```

As you can see above, the cert is only generated for a year. It has been created on PaSSHport automated installation.

Renew certificate with OpenSSL

To renew the certificate, use the openssl command, as follow :

```
root@passhport:~# openssl req -new -key "/home/passhport/certs/key.pem" \
-config "/home/passhport/passhport/tools/openssl-for-passhportd.cnf" \
-out "/home/passhport/certs/cert.pem" \
-subj "/C=FR/ST=Ile De France/L=Ivry sur Seine/O=LibrIT/OU=DSI/CN=passhport.librit.fr
↪" \
-x509 \
-days 365 \
-sha256 \
-extensions v3_req
root@passhport:~#
```

This will generate a self-signed certificate, like the one generated during the installation. It will be valid for 1 year. Change the values to your needs.

Restart passhportd

You now just need to restart passhportd :

```
root@passhport:~# systemctl restart passhportd.service
root@passhport:~#
```

You should now be able to use PaSSHport again.

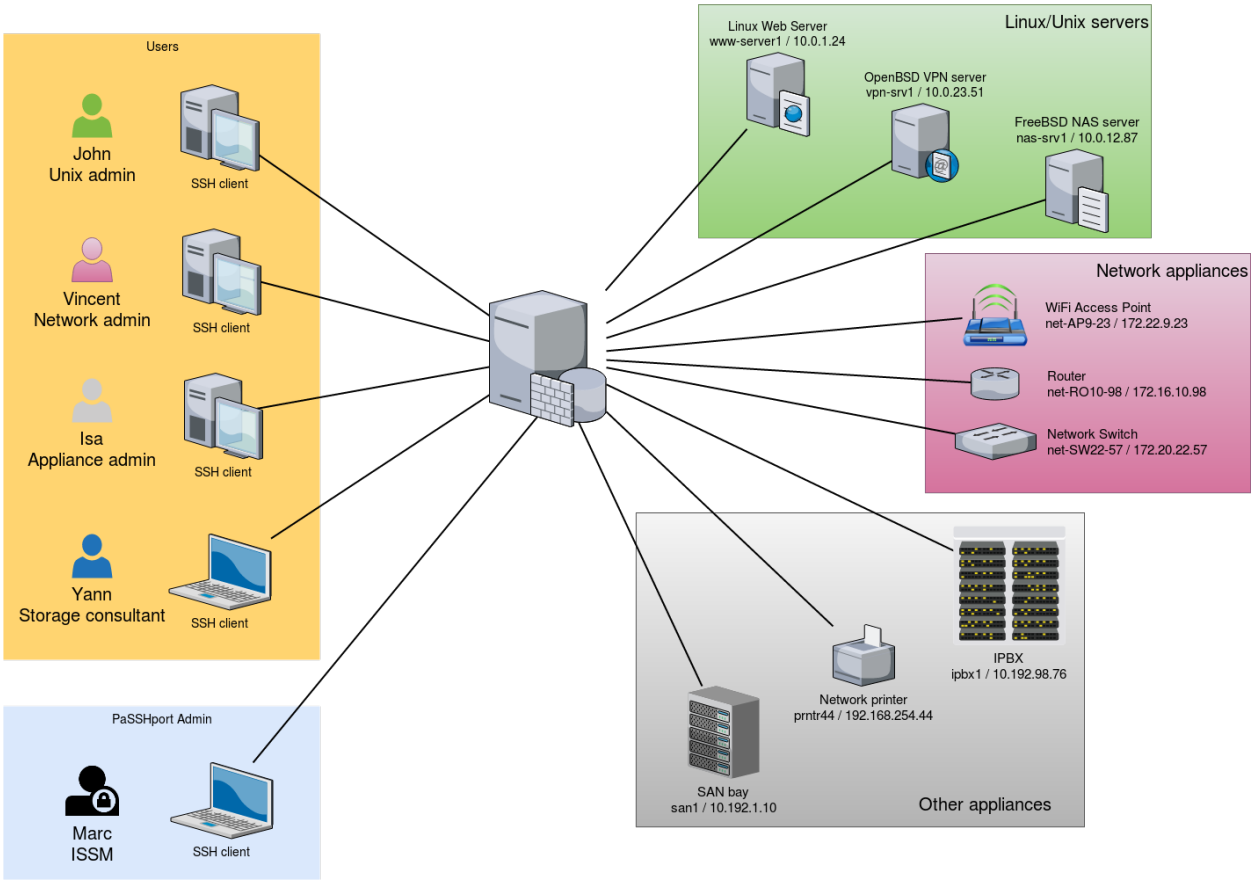
1.3 Getting started

So you now have a brand new installation of PaSSHport, but you don't know what to do next...

1.3.1 Example prerequisites

For this tutorial, we will use the following infos :

PaSSHport example architecture



- 1 PaSSHport node

We'll use a monolithic installation of PaSSHport : passhportd, passhport and passhport-admin are on the same host.

3 users

- John, a linux/unix administrator, who needs to access all linux/unix servers
- Vincent, a network administrator, who needs to access all network appliances
- Alice, a general appliance administrator who needs to access all tier appliances
- Yann, a consultant who's here for a temporary mission about storage infrastructure, that need to access the NAS server and a the SAN bay

1 PaSSHport admin

- Marc, the ISSM, who configures PaSSHport, to control all the access rights

3 types of targets

- Linux/Unix servers :
- 1 web server, Linux, www-server / 10.0.1.24
- 1 VPN server, OpenBSD, vpn-srv1 / 10.0.23.51
- 1 NAS server, FreeBSD, nas-srv1 / 10.0.12.87
- Network appliances
- 1 WiFi access points, net-AP9-23 / 172.22.9.23

- 1 router, net-RO10-98 / 172.16.10.98
- 1 network switch, net-SW22-57 / 172.20.22.57
- Other appliances
- 1 IPBX, ipbx1 / 10.192.98.76
- 1 Network printer, prntr44 / 192.168.254.44
- 1 SAN bay, san1 / 10.192.1.10

1.3.2 Configure targets

First of all, we'll include the targets into PaSSHport.

Let's connect to your PaSSHport node, and add the linux target. We can do this as passhport user :

```
passhport@passhport-server:~$ passhport-admin target create www-server 10.0.1.24
OK: "www-server" -> created
passhport@passhport-server:~$
```

We can check that the target has been well recorded :

```
passhport@passhport-server:~$ passhport-admin target list
www-server
passhport@passhport-server:~$
```

Now let's add the other Linux/Unix server :

```
passhport@passhport-server:~$ passhport-admin target create vpn-srv1 10.0.23.51
OK: "vpn-srv1" -> created
passhport@passhport-server:~$ passhport-admin target create nas-srv1 10.0.23.51
OK: "nas-srv1" -> created
passhport@passhport-server:~$
```

Do the same for the network appliances, and the remaining :

```
passhport@passhport-server:~$ passhport-admin target create net-AP9-23 172.22.9.23
OK: "net-AP9-23" -> created
passhport@passhport-server:~$ passhport-admin target create net-RO10-98 172.16.10.98
OK: "net-RO10-98" -> created
passhport@passhport-server:~$ passhport-admin target create net-SW22-57 172.20.22.57
OK: "net-SW22-57" -> created
passhport@passhport-server:~$ passhport-admin target create ipbx1 10.192.98.76
OK: "ipbx1" -> created
passhport@passhport-server:~$ passhport-admin target create prntr44 192.168.254.44
OK: "prntr44" -> created
passhport@passhport-server:~$ passhport-admin target create san1 10.192.1.10
OK: "san1" -> created
passhport@passhport-server:~$
```

We now have all our targets configured into PaSSHport.

1.3.3 Special target, with a specific login

We want to be able to connect to the SAN bay, as another user, because Yann should not have access to the SAN bay as the root user, but as "admin" user :

```
root@passhport-server:~# passhport-admin target create
Name: admin@san1
Hostname: 10.192.1.10
Login (default is root): admin
Port: 22
SSH Options:
Comment: SAN bay, login as admin user, not root.
OK: "admin@san1" -> created
root@passhport-server:~#
```

The SAN will now be accessible through two targets : "san1" and "admin@san1".

1.3.4 Configure target's groups

We'll group the targets we just created into three groups : unices, network and others.

We create the groups :

```
passhport@passhport-server:~$ passhport-admin targetgroup create unices
OK: "unices" -> created
passhport@passhport-server:~$ passhport-admin targetgroup create network
OK: "network" -> created
passhport@passhport-server:~$ passhport-admin targetgroup create others
OK: "others" -> created
passhport@passhport-server:~$
```

Now we put the targets into the corresponding target groups :

```
passhport@passhport-server:~$ passhport-admin targetgroup addtarget www-server unices
OK: "www-server" added to "unices"
passhport@passhport-server:~$
```

I'm a bit lazy, so I'll script the remainings :

```
passhport@passhport-server:~$ for UNICE in vpn-srv1 nas-srv1; do passhport-admin␣
↪targetgroup addtarget ${UNICE} unices; done
OK: "vpn-srv1" added to "unices"
OK: "nas-srv1" added to "unices"
passhport@passhport-server:~$ for NETAPPLIANCE in net-AP9-23 net-RO10-98 net-SW22-57;␣
↪do passhport-admin targetgroup addtarget ${NETAPPLIANCE} network; done
OK: "net-AP9-23" added to "network"
OK: "net-RO10-98" added to "network"
OK: "net-SW22-57" added to "network"
passhport@passhport-server:~$ for OTHERAPPLIANCE in ipbx1 prntr44 san1; do passhport-
↪admin targetgroup addtarget ${OTHERAPPLIANCE} others; done
OK: "ipbx1" added to "others"
OK: "prntr44" added to "others"
OK: "san1" added to "others"
passhport@passhport-server:~$
```

We'll create a last group, that will have all the targets in it (again, I'm gonna script this) :

```
passhport@passhport-server:~$ passhport-admin targetgroup create all-targets
OK: "all-targets" -> created
passhport@passhport-server:~$ for TARGET in `passhport-admin target list`; do␣
↪passhport-admin targetgroup addtarget ${TARGET} all-targets; done
```

(continues on next page)

(continued from previous page)

```
OK: "ipbx1" added to "all-targets"
OK: "nas-srv1" added to "all-targets"
OK: "net-AP9-23" added to "all-targets"
OK: "net-RO10-98" added to "all-targets"
OK: "net-SW22-57" added to "all-targets"
OK: "prntr44" added to "all-targets"
OK: "san1" added to "all-targets"
OK: "vpn-srv1" added to "all-targets"
OK: "www-server" added to "all-targets"
passhport@passhport-server:~$
```

We're now done with the targets/targetgroups, at least for the moment...

1.3.5 Configure the users

We take it for granted that our users have all created a ssh public key (rsa, dsa or ecdsa), and that they gave us the public part. We have all the following keys :

Alice, a 2048 bits RSA key :

```
ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQC8JMsMgyRUEMoq31rPTIWpWKgGFQ7fxt5Kray8yzCPga2pohMLstjJeHpWjkVhH8FhRUv
↪V/42j3izeRH5liXFwotxzfpqTijTxAfj/
↪60IadcUSf5dE8WAiREarrV82ieU5eNZ4FoCH4W0xPS8pEYJDv6hQ8TFHYQCrwHloA3HgzEJgQSFWaS3niMDfNbgbJEOVhXuT2l7
↪tp+9FEAqGCH3kTuFhFnWCgguQxDxH4XiIj7n2w79ARPzMbn2vTtd+6N0or7 alice@myfirm.com
```

John, an 521 bits ECDSA key :

```
ecdsa-sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHTlnhl23T9NiHn06wWaDpT1aJqEY0aOW7E4dfu7kQJsmF
↪yJYbKwwPQEaHpiQoHMaBfsgA2BYS5cNVcrOpLk8nHgKSJGEcdYipbZZxqDrLaeX3lBA== john@myfirm.
↪com
```

Marc, a 4096 bits RSA key :

```
ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDFOU5Saf+epkm79BeSniE7VtYMexJeL6BvXUsKUb7m8W4gnD3YTBW93uykO/
↪6ovi9TfYdm+4nKQ9gUGUgzNyD8o7zW8w6wKogoL24UbJKmkZOCU1IgHJSt1QYIs/qHQZ2MR6S6K2f/
↪1J1joYINPtGpQJ475OZfYQbP79fEdRdylupC8L+fvxkka4C0Uxj0I1VjDCVJCjO0md5oXzN75I2aw+RFWuiiL5P/
↪gHRu+2iff2rdhebJZs4ux8u76LQLzYsG9a85Xlagw6N7/aXWnUZ/9gqoF/
↪qVUHfS8ggesTwEJyNnY7EpPcKRUcwnlonn5CIS++Yo8iqjLd93RjFxShUqXlw9Cct4hdh/clW/
↪QYsJRMfN9860mZ9v9dEitM2X1w8HCCD5NAHGqRRrtONM99kZRxmkCQ/
↪Tb+jXvJ+VAl4qffuPPdxY+Bev7wygm4rVnjF2Ac5ioWb4Zd+zIb712VTQDQlRxsu73yWtHSodeSgPpgCWTjCwW/
↪841QbPGkclnE6DKIwQ/
↪vxC0ggSXouc5G6j0gHu90eQ24XL6Gurqr2C11w9saRyzrYRRlS0Ihkp3rMSteVcvrb1Qi4UGmJCHHSBhvP8jRFH4mbdkSGyzsxt
↪60fdEELQyX+kNFQ2VoCw== marc@myfirm.com
```

Vincent, a 521 bits ECDSA key :

```
ecdsa-sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHJk+qDLEi283+rUmSek3eEF4PqXYMmQlPTj352w0XO75P
↪xVc+ypwOb2vv6pcjVsvuHTwHgXR2ElyfE8gGV7mITyXMdDyoWP5N8Ly3s7njNChSL9z3NiG38lg3E4Vg10nbmmoZZCA3WCffV4u
↪vincent@myfirm.com
```

And Yann, a 2048 bits RSA key :

```
ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com
```

With those keys, we can add the users as follow…

- Interactively :

```
passhport@passhport-server:~$ passhport-admin user create
Email (user name): alice@myfirm.com
SSH Key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQC8JMsMgyRUEMoq31rPTIWpWKgGFQ7fxt5Kray8yzCPga2pohMLstjJeHpWjkVhH8FhRUv
↪V/42j3izeRH5liXFwotxzfpqTijTxAfj/
↪60IadcUSf5dE8WAiREarrV82ieU5eNZ4FoCH4W0xPS8pEYJDv6hQ8TFHYQCrwHloA3HgzEJgQSFWaS3niMDfNbgbJEOVhXuT2l7
↪tp+9FEAqGCH3kTuFhFnWCgguQxDxH4XiIj7n2w79ARPzMbn2vTtd+6N0or7 alice@myfirm.com
Comment: Alice is the general applicance admin
OK: "alice@myfirm.com" -> created
passhport@passhport-server:~$
```

- On a single line, one shot :

```
passhport@passhport-server:~$ passhport-admin user create john@myfirm.com "ecdsa-sha2-
↪nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHTlnhl23T9NiHn06wWaDpT1aJqEY0aOW7E4dfu7kQJsmH
↪yJYbKwwPQEaHpiQoHMaBfsgA2BYS5cNVcrOpLk8nHgKSJGEcdYipbZZxqDrLaeX3lBA== john@myfirm.
↪com" --comment="John is the Unices admin. He rocks."
OK: "john@myfirm.com" -> created
passhport@passhport-server:~$
```

We add the others :

```
passhport@passhport-server:~$ passhport-admin user create marc@myfirm.com "ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDFOU5Saf+epkm79BeSniE7VtYMexJeL6BvXUsKUb7m8W4gnD3YTBW93uykO/
↪6ovi9TfYdm+4nKQ9gUGUgzNyD8o7zW8w6wKogoL24UbJKmkZOCU1IgHJSt1QYIs/qHQZ2MR6S6K2f/
↪1J1joYINPtGpQJ475OZfYQbP79fEdRdylupC8L+fvxkka4C0Uxj0I1VjDCVJCjO0md5oXzN75I2aw+RFWuiiL5P/
↪gHRu+2iff2rdhebJZs4ux8u76LQLzYsG9a85Xlagw6N7/aXWnUZ/9gqoF/
↪qVUHfS8ggesTwEJyNnY7EpPcKRUcwnlonn5CIS++Yo8iqjLd93RjFxShUqXlw9Cct4hdh/clW/
↪QYsJRMfN9860mZ9v9dEitM2X1w8HCCD5NAHGqRRrtONM99kZRxmkCQ/
↪Tb+jXvJ+VAl4qffuPPdxY+Bev7wygm4rVnjF2Ac5ioWb4Zd+zIb712VTQDQlRxsu73yWtHSodeSgPpgCWTjCwW/
↪841QbPGkclnE6DKIwQ/
↪vxC0ggSXouc5G6j0gHu90eQ24XL6Gurqr2C11w9saRyzrYRRlS0Ihkp3rMSteVcvrb1Qi4UGmJCHHSBhvP8jRFH4mbdkSGyzsxt
↪60fdEELQyX+kNFQ2VoCw== marc@myfirm.com"
OK: "marc@myfirm.com" -> created
passhport@passhport-server:~$ passhport-admin user create vincent@myfirm.com "ecdsa-
↪sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHJk+qDLEi283+rUmSek3eEF4PqXYMmQlPTj352w0XO75E
↪xVc+ypwOb2vv6pcjVsvuHTwHgXR2ElyfE8gGV7mITyXMdDyoWP5N8Ly3s7njNChSL9z3NiG38lg3E4Vg10nbmmoZZCA3WCffV4u
↪vincent@myfirm.com" --comment="Vincent is the network admin."
OK: "vincent@myfirm.com" -> created
passhport@passhport-server:~$ passhport-admin user create yann@otherfirm.com "ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com" --comment="Yann is an external consultant, for a temporary␣
↪mission bout storage infrastructure."
```

(continues on next page)

(continued from previous page)

```
OK: "yann@otherfirm.com" -> created
passhport@passhport-server:~$
```

As you can see above, I forgot to put a comment on "marc@myfirm.com" account. Let's add one :

```
passhport@passhport-server:~$ passhport-admin user edit marc@myfirm.com --newcomment=
↪"Marc is the ISSM. He access all."
OK: "marc@myfirm.com" -> edited
passhport@passhport-server:~$
```

Users are now created. Let's put them in usergroups...

1.3.6 Configure usergroups :

Even if in this example we only have one user for each purpose of administration, it's generaly a good idea to add a group for one type skill.

Let's add those groups :

```
passhport@passhport-server:~# passhport-admin usergroup create unices_admins
OK: "unices_admins" -> created
passhport@passhport-server:~# passhport-admin usergroup create network_admins
OK: "network_admins" -> created
passhport@passhport-server:~# passhport-admin usergroup create appliance_admins
OK: "appliance_admins" -> created
passhport@passhport-server:~# passhport-admin usergroup create super_admins
OK: "super_admins" -> created
passhport@passhport-server:~#
```

We add the users to each corresponding groups :

```
passhport@passhport-server:~$ passhport-admin usergroup adduser john@myfirm.com␣
↪unices_admins
OK: "john@myfirm.com" added to "unices_admins"
passhport@passhport-server:~$ passhport-admin usergroup adduser vincent@myfirm.com␣
↪network_admins
OK: "vincent@myfirm.com" added to "network_admins"
passhport@passhport-server:~$ passhport-admin usergroup adduser alice@myfirm.com␣
↪appliance_admins
OK: "alice@myfirm.com" added to "appliance_admins"
passhport@passhport-server:~$ passhport-admin usergroup adduser marc@myfirm.com super_
↪admins
OK: "marc@myfirm.com" added to "super_admins"
passhport@passhport-server:~$
```

1.3.7 Connect usergroups and targetgroups :

We now can connect each usergroups to targetgroups :

```
passhport@passhport-server:~# passhport-admin targetgroup addusergroup unices_admins␣
↪unices
OK: "unices_admins" added to "unices"
passhport@passhport-server:~# passhport-admin targetgroup addusergroup network_admins␣
↪network
```

(continues on next page)

(continued from previous page)

```
OK: "network_admins" added to "network"
passhport@passhport-server:~# passhport-admin targetgroup addusergroup appliance_
↪admins others
OK: "appliance_admins" added to "others"
passhport@passhport-server:~# passhport-admin targetgroup addusergroup super_admins␣
↪all-targets
OK: "super_admins" added to "all-targets"
passhport@passhport-server:~#
```

1.3.8 Special configuration for Yann :

Because Yann is only here for a short mission, and need to access to different targets, that won't be grouped into a targetgroup, so we connect him directly to the targets :

```
passhport@passhport-server:~$ passhport-admin target adduser yann@otherfirm.com nas-
↪srv1
OK: "yann@otherfirm.com" added to "nas-srv1"
passhport@passhport-server:~# passhport-admin target adduser yann@otherfirm.com␣
↪admin@san1
OK: "yann@otherfirm.com" added to "admin@san1"
passhport@passhport-server:~#
```

As you can see above, we did not give Yann access directly to san1 as root, but as admin user, through the admin@san1 target we created before.

1.3.9 Check rights :

We can check what we configured with the "show" sub-command of passhport-admin :

```
passhport@passhport-server:~$ passhport-admin user show marc@myfirm.com
Email: marc@myfirm.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDFOU5Saf+epkm79BeSniE7VtYMexJeL6BvXUsKUb7m8W4gnD3YTBW93uykO/
↪6ovi9TfYdm+4nKQ9gUGUgzNyD8o7zW8w6wKogoL24UbJKmkZOCU1IgHJSt1QYIs/qHQZ2MR6S6K2f/
↪1J1joYINPtGpQJ475OZfYQbP79fEdRdylupC8L+fvxkka4C0Uxj0I1VjDCVJCjO0md5oXzN75I2aw+RFWuiiL5P/
↪gHRu+2iff2rdhebJZs4ux8u76LQLzYsG9a85Xlagw6N7/aXWnUZ/9gqoF/
↪qVUHfS8ggesTwEJyNnY7EpPcKRUcwnlonn5CIS++Yo8iqjLd93RjFxShUqXlw9Cct4hdh/clW/
↪QYsJRMfN9860mZ9v9dEitM2X1w8HCCD5NAHGqRRrtONM99kZRxmkCQ/
↪Tb+jXvJ+VAl4qffuPPdxY+Bev7wygm4rVnjF2Ac5ioWb4Zd+zIb712VTQDQlRxsu73yWtHSodeSgPpgCWTjCwW/
↪841QbPGkclnE6DKIwQ/
↪vxC0ggSXouc5G6j0gHu90eQ24XL6Gurqr2C11w9saRyzrYRRlS0Ihkp3rMSteVcvrb1Qi4UGmJCHHSBhvP8jRFH4mbdkSGyzsxt
↪60fdEELQyX+kNFQ2VoCw== marc@myfirm.com
Comment: Marc is the ISSM. He access all.
Accessible target list: ipbx1 nas-srv1 net-AP9-23 net-RO10-98 net-SW22-57 prntr44␣
↪san1 vpn-srv1 www-server

Details in access:
Accessible directly:
Accessible through usergroups:
super_admins: www-server ; vpn-srv1 ; nas-srv1 ; net-AP9-23 ; net-RO10-98 ; net-SW22-
↪57 ; ipbx1 ; prntr44 ; san1 ;
Accessible through targetgroups:
passhport@passhport-server:~$
```

As you can see, the "show" sub-command shows how the user has access to each target. We can see above that Marc has access to all the target we configured, because we placed him in the "super_admins" group.

Here is the example for Yann :

```
passhport@passhport-server:~$ passhport-admin user show yann@otherfirm.com
Email: yann@otherfirm.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com
Comment: Yann is an external consultant, for a temporary mission bout storage␣
↪infrastructure.
Accessible target list: nas-srv1 san1

Details in access:
Accessible directly: nas-srv1 ; san1 ;
Accessible through usergroups:
Accessible through targetgroups:
passhport@passhport-server:~$
```

You can see above that Yann has a direct access to targets, not through usergroups, or targetgroups.

1.3.10 Let's connect !

Let's say that I'm John, I connect to PaSSHport, using the id_rsa key that I sent to the PaSSHport admin :

```
john@my-desktop:~$ ssh passhport@passhport-server
Welcome john@myfirm.com.
Here is the list of servers you can access:
1  www-server  10.0.1.24
2  vpn-srv1    10.0.23.51
3  nas-srv1    10.0.12.87
Type the number, name or hostname of the server you want to connect to :
```

As John, I can see that I can access to 3 servers : www-server, vpn-srv1 and nas-srv1. I can now access to each server, using :

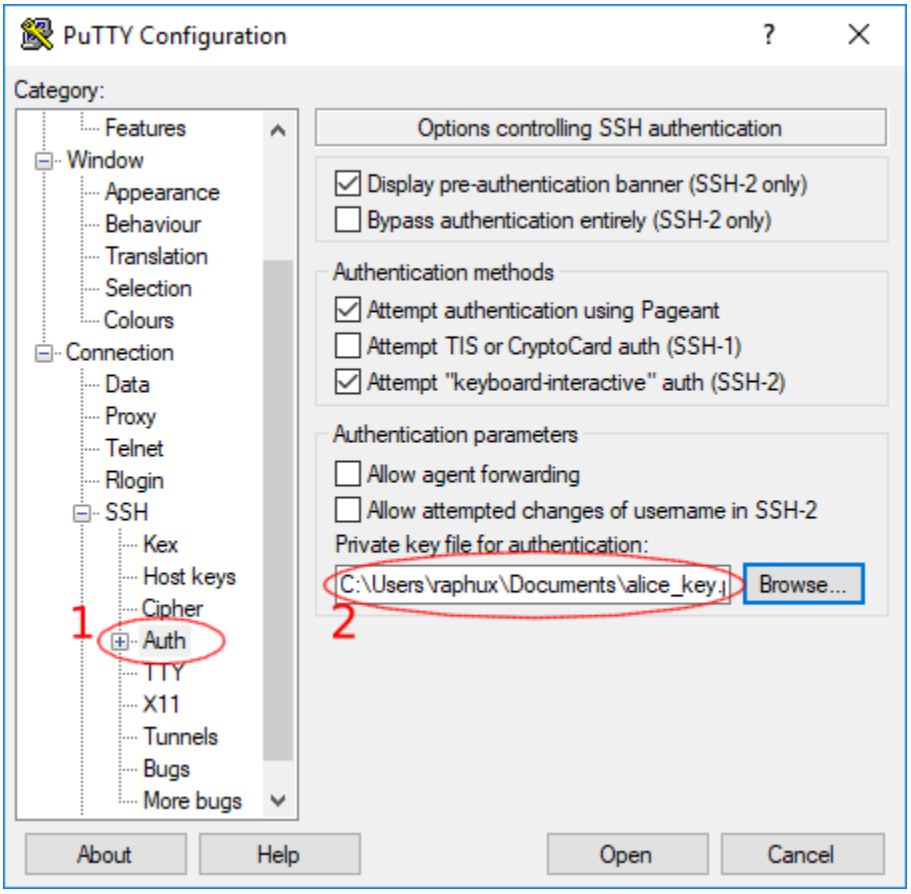
- the number in the first column;
- the name of the server (www-serve...);
- the IP address.

```
john@my-desktop:~$ ssh passhport@passhport-server
Welcome john@myfirm.com.
Here is the list of servers you can access:
1  www-server  10.0.1.24
2  vpn-srv1    10.0.23.51
3  nas-srv1    10.0.12.87
Type the number, name or hostname of the server you want to connect to : 1
Linux www-server 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64
root@www-server:~#
```

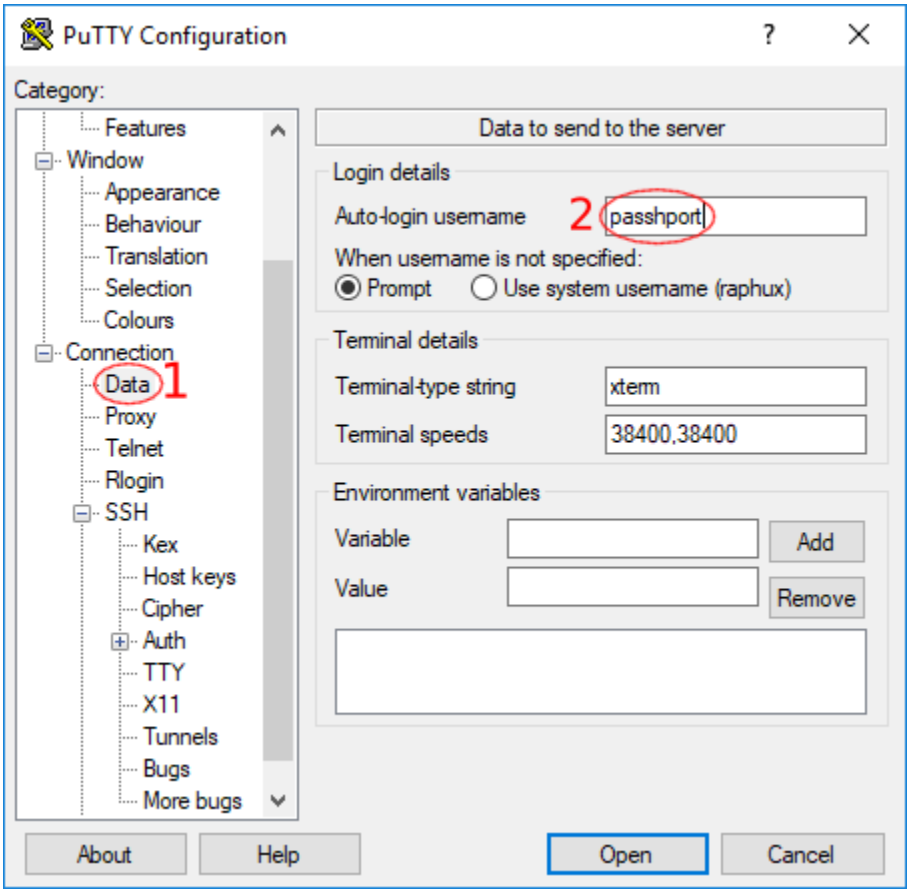
John is now on the www-server.

Let's say that I'm now Alice, a Windows user. I'm going to use putty to connect to PaSSHport. Let's configure putty...

We launch Putty (you can download it from here), and on the left configuration tree, goes to *Connection* -> *SSH* -> *Auth* , then select the ppk key Alice generated (with puttygen for exemple) :



Then we go to *Connection* -> *SSH* -> *Data* , and set the login name as passhport :

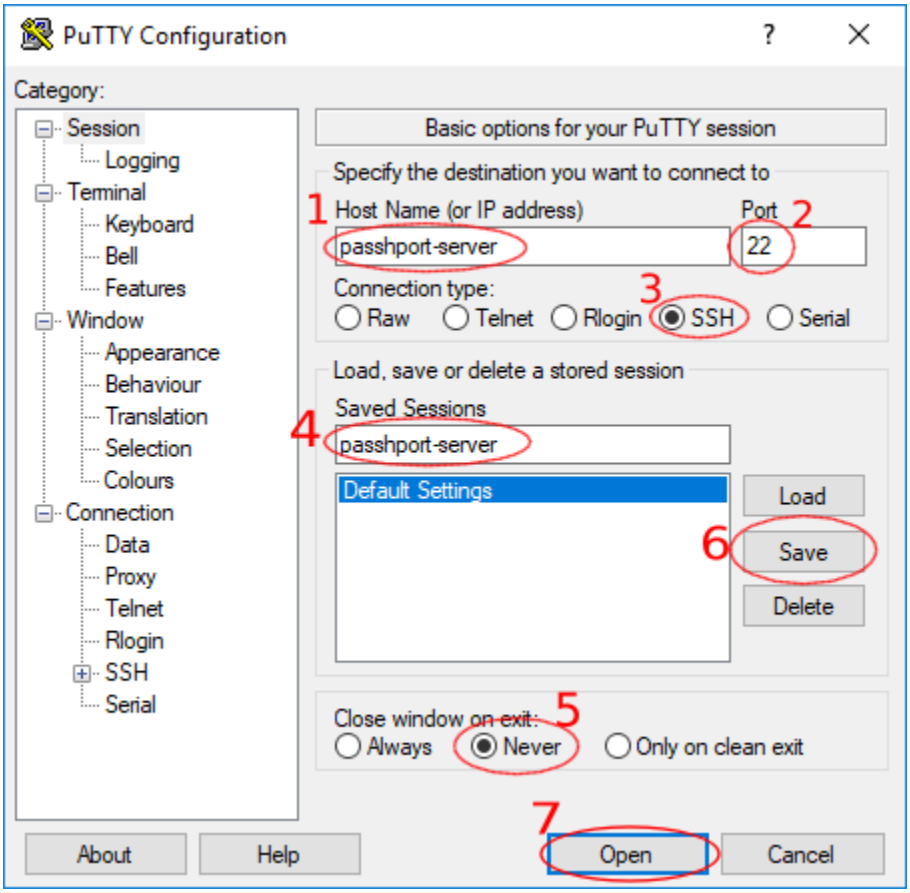


Finally, we go to the root of the configuration tree *Session* :

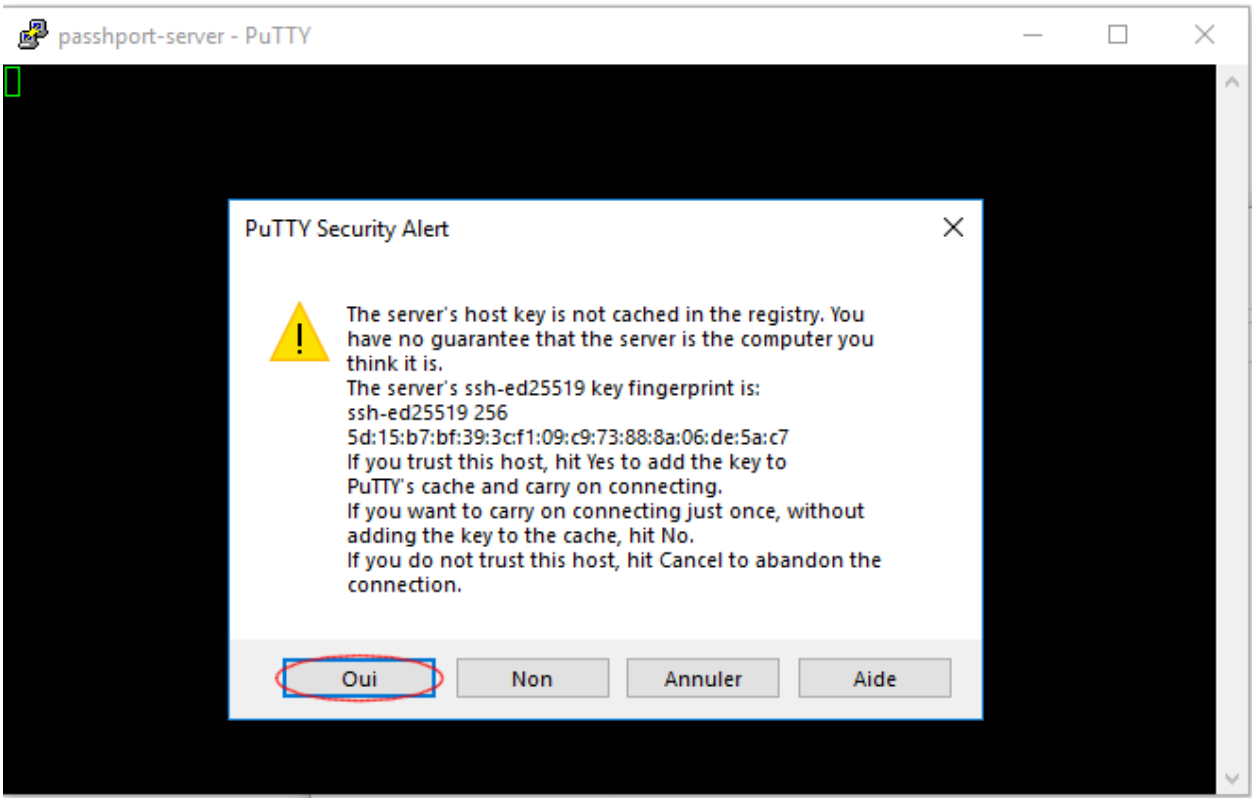
- enter the hostname or IP of your PaSSHport server
- enter its SSH port (usually 22)
- select *SSH* as connection type
- enter a name for this connection configuration

For debugging purpose, it may be useful to *Never* close window on exit (so you can see the error message).

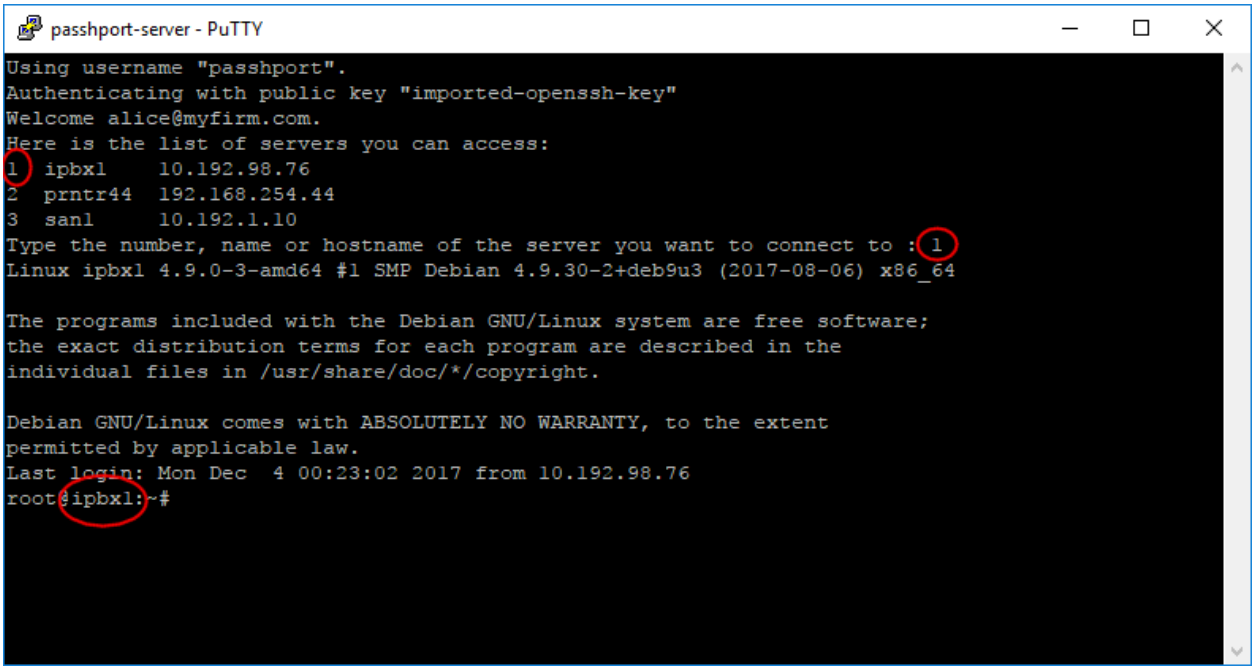
Save, and click Open !



If it's the first time we connect to the PaSSHport server, we'll have a window that says the fingerprint is new, and ask us if we want to accept it… Just accept it :



Then we'll have the PaSSHport prompt, and as we want to connect to IPBX, we select *1* :



We are now landed on our target.

Last relevant example, Yann, who access nas-srv1 and san1. He uses a linux laptop :

```
yann@my-laptop:~$ ssh passhport@passhport-server
```

(continues on next page)

(continued from previous page)

```
Welcome yann@otherfirm.com.
Here is the list of servers you can access:
1  nas-srv1    10.0.12.87
2  admin@san1  10.192.1.10  SAN bay, login as admin user, not root.
Type the number, name or hostname of the server you want to connect to :
```

He can now connect to any of those two servers.

1.3.11 Delete a user

Yann has finished his mission, and left the compagny. There is two way to revoke his access :

- remove all his target ;
- delete the user.

You may prefer the first way if you know that Yann will may come back later to do another mission, so you won't have to recreate the user (get his ssh key, etc...). Here is how you can delete his access...

First, list his rights :

```
passhport@passhport-server:~$ passhport-admin user show yann@otherfirm.com
Email: yann@otherfirm.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com
Comment: Yann is an external consultant, for a temporary mission bout storage␣
↪infrastructure.
Accessible target list: admin@san1 nas-srv1

Details in access:
Accessible directly: nas-srv1 ; admin@san1 ;
Accessible through usergroups:
Accessible through targetgroups:
passhport@passhport-server:~$
```

You can see that he has access to nas-srv1 and admin@san1, directly. Let's revoke those access :

```
passhport@passhport-server:~$ passhport-admin target rmuser yann@otherfirm.com␣
↪admin@san1
OK: "yann@otherfirm.com" removed from "admin@san1"
passhport@passhport-server:~$ passhport-admin target rmuser yann@otherfirm.com nas-
↪srv1
OK: "yann@otherfirm.com" removed from "nas-srv1"
passhport@passhport-server:~$
```

Yann won't have access to any target anymore :

```
passhport@passhport-server:~$ passhport-admin user show yann@otherfirm.com
Email: yann@otherfirm.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com
```

(continues on next page)

(continued from previous page)

```
Comment: Yann is an external consultant, for a temporary mission bout storage␣
↪infrastructure.
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
passhport@passhport-server:~$
```

The second option, is to delete the user :

```
passhport@passhport-server:~$ passhport-admin user delete yann@otherfirm.com
Email: yann@otherfirm.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@otherfirm.com
Comment: Yann is an external consultant, for a temporary mission bout storage␣
↪infrastructure.
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
Are you sure you want to delete yann@otherfirm.com? [y/N] y
OK: "yann@otherfirm.com" -> deleted
passhport@passhport-server:~$
```

1.3.12 Conclusion

You should now be able to use the basic functions of PaSSHport.

1.4 passhport-admin usage

This chapter describes the usage of passhport-admin

One chapter per sub-module :

1.4.1 CLI usage

You can call passhport-admin's CLI, by calling `passhport-admin` with the `-i`.

You can then configure passhportd through this command line. Refer to the good section in the submodule doc to know how to use them.

1.4.2 user

Usages :

```
passhport-admin user list
passhport-admin user search [<pattern>]
passhport-admin user show [<name>]
passhport-admin user create [((<name> <sshkey>) [--comment=<comment>])]
passhport-admin user edit [(<name> [--newname=<name>] [--newsshkey=<sshkey>] [--
↪newcomment=<comment>])]
passhport-admin user delete [([-f | --force] <name>)]
```

list

passhport-admin user list show all the configured users.

Example :

```
admin@bastion:~$ passhport-admin user list
admin1@compagny.com
admin2@compagny.com
alice@compagny.com
bob@compagny.com
admin@bastion:~$
```

search

passhport-admin user search [<PATTERN>] searches in the user list for all user that correspond to the given pattern.

Example :

```
admin@bastion:~# passhport-admin user search admin
admin1@compagny.comi
admin2@compagny.com
admin@bastion:~#
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin user search
Pattern: alice
alice@compagny.com
admin@bastion:~#
```

show

passhport-admin user show <*NAME*> shows informations about the user <NAME>.

Example :

```
admin@bastion:~# passhport-admin user show alice@compagny.com
Email: alice@compagny.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDFOU5Saf+epkm79BeSniE7VtYMexJeL6BvXUsKUb7m8W4gnD3YTBW93uykO/
↪6ovi9TfYdm+4nKQ9gUGUgzNyD8o7zW8w6wKogoL24UbJKmkZOCU1IgHJSt1QYIs/qHQZ2MR6S6K2f/
↪1J1joYINPtGpQJ475OZfYQbP79fEdRdylupC8L+fvxkka4C0Uxj0I1VjDCVJCjO0md5oXzN75I2aw+RFWuiiL5P/
↪gHRu+2iff2rdhebJZs4ux8u76LQLzYsG9a85Xlagw6N7/aXWnUZ/9gqoF/
↪qVUHf8[illegible]cKRUcwnlonn5CIS++Yo8iqjLd93RjFxShUqXlw9Cct4hdh/clW/
↪QYsJRMfN9860mZ9v9dEitM2X1w8HCCD5NAHGqRRrtONM99kZRxmkCQ/
↪Tb+jXvJ+VAl4qffuPPdxY+Bev7wygm4rVnjF2Ac5ioWb4Zd+zIb712VTQDQlRxsu73yWtHSodeSgPpgCWTjCwW/
↪841QbPGkclnE6DKIwQ/
↪vxC0qgSXouc5G6j0gHu90eO24XL6Gurgr2C11w9saRvzrYRRlS0Ihkp3rMSteVcyrb1Oj4UGmJCHHSBhyP8jRFH4mbdkSGyzsxt
```

(continues on next page)

(continued from previous page)

```
Comment:
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
admin@bastion:~#
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin user show
Name: alice@compagny.com
Email: alice@compagny.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDFOU5Saf+epkm79BeSniE7VtYMexJeL6BvXUsKUb7m8W4gnD3YTBW93uykO/
↪6ovi9TfYdm+4nKQ9gUGUgzNyD8o7zW8w6wKogoL24UbJKmkZOCU1IgHJSt1QYIs/qHQZ2MR6S6K2f/
↪1J1joYINPtGpQJ475OZfYQbP79fEdRdylupC8L+fvxkka4C0Uxj0I1VjDCVJCjO0md5oXzN75I2aw+RFWuiiL5P/
↪gHRu+2iff2rdhebJZs4ux8u76LQLzYsG9a85Xlagw6N7/aXWnUZ/9gqoF/
↪qVUHfS8ggesTwEJyNnY7EpPcKRUcwnlonn5CIS++Yo8iqjLd93RjFxShUqXlw9Cct4hdh/clW/
↪QYsJRMfN9860mZ9v9dEitM2X1w8HCCD5NAHGqRRrtONM99kZRxmkCQ/
↪Tb+jXvJ+VAl4qffuPPdxY+Bev7wygm4rVnjF2Ac5ioWb4Zd+zIb712VTQDQlRxsu73yWtHSodeSgPpgCWTjCwW/
↪841QbPGkclnE6DKIwQ/
↪vxC0ggSXouc5G6j0gHu90eQ24XL6Gurqr2C11w9saRyzrYRRlS0Ihkp3rMSteVcvrb1Qi4UGmJCHHSBhvP8jRFH4mbdkSGyzsxt
↪60fdEELQyX+kNFQ2VoCw== alice@compagny.com
Comment:
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
admin@bastion:~#
```

create

passhport-admin user create [((<name> <sshkey>) [–comment=<comment>])] creates a new user.

Argument	Description
<name>	Name of the target to edit
<sshkey>	SSH key of the user (use " as the below example)
–comment	Comment concerning the user (optional)

Example :

```
admin@bastion:~$ passhport-admin user create bob@compagny.com "ecdsa-sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHTlnhl23T9NiHn06wWaDpT1aJqEY0aOW7E4dfu7kQJsmE
↪yJYbKwwPQEaHpiQoHMaBfsgA2BYS5cNVcrOpLk8nHgKSJGEcdYipbZZxqDrLaeX3lBA== bob@mydesktop"
OK: "bob@compagny.com" -> created
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin user create bob@compagny.com "ecdsa-sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHTlnhl23T9NiHn06wWaDpT1aJqEY0aOW7E4dfu7kQJsmH
↪yJYbKwwPQEaHpiQoHMaBfsgA2BYS5cNVcrOpLk8nHgKSJGEcdYipbZZxqDrLaeX3lBA== bob@mydesktop"
OK: "bob@compagny.com" -> created
admin@bastion:~$ passhport-admin user create
Email (user name): john@ext-compagny.com
SSH Key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@mylaptop.com
Comment: John is a extern expert.
OK: "john@ext-compagny.com" -> created
admin@bastion:~$
```

edit

passhport-admin user edit [(<name> [–newname=<name>] [–newsshkey=<sshkey>] [–newcomment=<comment>])] edits an existing user.

Argument	Description
<name>	Name of the user to edit
–newname	New name of the user if you want to rename it (optional)
–newsshkey	New SSH key of the user (use " as the below example) (optional)
–newcomment	New comment concerning the user (optional)

Example :

```
admin@bastion:~$ passhport-admin user edit john@ext-compagny.com --newname=john.
↪doe@ext-compagny.com --newcomment="John is a extern expert, he'll be here until␣
↪january 18th."
OK: "john@ext-compagny.com" -> edited
admin@bastion:~$
```

If no argument is given, user enters in interactive mode. It firsts shows all parameters of the user, then displays each parameters for a change. User can keep any previous configured parameter, just by typing "Enter". They only exception is the comment. If user wants to remove the comment, he just type "Enter", and will then be asked if the original comment should be removed or not.

Example :

```
admin@bastion:~$ passhport-admin user edit
Name of the user you want to modify: john.doe@ext-compagny.com
Email: john.doe@ext-compagny.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@mylaptop.com
```

(continues on next page)

(continued from previous page)

```
Comment: John is a extern expert, he'll be here until january 18th.
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
New name:
New SSH key:
New comment: John is a extern expert, he'll be here until february 2nd
OK: "john.doe@ext-compagny.com" -> edited
admin@bastion:~$
```

As you can see above, we only changed the "New comment" entry. If an entry is simply replied with "enter", it keeps the previous value.

delete

passhport-admin user delete [([-f | –force] <name>)] delete a user.

Argument	Description
<name>	Name of the user to delete
-f or –force	If used, user won't be prompt for confirmation

Example :

```
admin@bastion:~$ passhport-admin user delete john.doe@ext-compagny.com
Email: john.doe@ext-compagny.com
SSH key: ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABAQCs9YpOfP9vgViYa1SSntrydEBLGyWGAr9nvEjqHcMwHQb9JEmhIjvk1ctb8+Kns3/
↪52F0hBrxic6k6UPvvvjbtJX33muFv5dd0k1W4lLcYe4ONTFwLOqCph4Is5r9lbZ5KXxhN/8YC/
↪08jBJow0CoYdc+Yr7MlA51+tEQFwPbuB5vHMUteye0IgmaH9MLzXes/j5BUhnBjDscWVQSvNHY4/
↪PKtHvIdvoI1uKAplstuHI6CDqnb0aJ5P9wME3P1lhRwcVDTm48/
↪AMcfmpp5s+DwOmyDGfGXf+hE0cu7ulAkwHBhR6ciJJg1pz4DqraglxyVyrt+PFq6KDeV/7WwoNEP␣
↪yann@mylaptop.com
Comment: John is a extern expert, he'll be here until february 2nd
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
Are you sure you want to delete john.doe@ext-compagny.com? [y/N] y
OK: "john.doe@ext-compagny.com" -> deleted
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin user delete
Name: bob@compagny.com
Email: bob@compagny.com
SSH key: ecdsa-sha2-nistp521␣
↪AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAHTlnhl23T9NiHn06wWaDpT1aJqEY0aOW7E4dfu7kQJsmE
↪yJYbKwwPQEaHpiQoHMaBfsgA2BYS5cNVcrOpLk8nHgKSJGEcdYipbZZxqDrLaeX3lBA== bob@mydesktop
```

(continued from previous page)

(continued from previous page)

```
Comment:
Accessible target list:

Details in access:
Accessible directly:
Accessible through usergroups:
Accessible through targetgroups:
Are you sure you want to delete bob@compagny.com? [y/N] y
OK: "bob@compagny.com" -> deleted
admin@bastion:~$
```

1.4.3 target

Usages :

```
passhport-admin target list
passhport-admin target search [<pattern>]
passhport-admin target checkaccess [<pattern>]
passhport-admin target show [<name>]
passhport-admin target create [((<name> <hostname>) [--login=<login>] [--type=<ssh>]␣
↪[--comment=<comment>] [--sshoptions=<sshoptions>] [--port=<port>])]
passhport-admin target edit [(<name> [--newname=<name>] [--newhostname=<hostname>] [--
↪newlogin=<login>] [--newcomment=<comment>] [--newsshoptions=<sshoptions>] [--
↪newport=<port>])]
passhport-admin target (adduser | rmuser) [(<username> <targetname>)]
passhport-admin target (addusergroup | rmusergroup) [(<usergroupname> <targetname>)]
passhport-admin target delete [([-f | --force] <name>)]
```

list

passhport-admin target list show all the configured targets.

Example :

```
admin@bastion:~$ passhport-admin target list
srv1.compagny.com
srv2.compagny.com
srv3.compagny.com
websrv.ext.client.com
webbackend.ext.client.com
admin@bastion:~$
```

search

passhport-admin target search [<PATTERN>] searches in the target list for all targets that correspond to the given pattern.

Example :

```
admin@bastion:~$ passhport-admin target search web
websrv.ext.client.com
webbackend.ext.client.com
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin target search
Pattern: web
websrv.ext.client.com
webbackend.ext.client.com
admin@bastion:~$
```

checkaccess

passhport-admin target checkaccess [<PATTERN>] verifies that PaSSHport has access to the all targets that correspond to the given pattern.

Example :

```
admin@bastion:~$ passhport-admin target checkaccess web
OK:    132.123.45.67   websrv.ext.client.com
OK:    132.234.56.78   webbackend.ext.client.com
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin target checkaccess
Pattern: web
OK:    132.123.45.67   websrv.ext.client.com
OK:    132.234.56.78   webbackend.ext.client.com
admin@bastion:~$
```

show

passhport-admin target show <*NAME*> shows informations about the target <NAME>.

Example :

```
admin@bastion:~$ passhport-admin target show websrv.ext.client.com
Name: websrv.ext.client.com
Hostname: 132.123.45.67
Server Type : ssh
Login: root
Port: 22
SSH options:
Comment:
Attached users:
Usergroup list:
Users who can access this target: admin1@compagny.com admin2@compagny.com
All usergroups:
Member of the following targetgroups: all-targets
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin target show
Name: websrv.ext.client.com
Name: websrv.ext.client.com
Hostname: 132.123.45.67
Server Type : ssh
Login: root
Port: 22
SSH options:
Comment:
Attached users:
Usergroup list:
Users who can access this target: admin1@compagny.com admin2@compagny.com
All usergroups:
Member of the following targetgroups: all-targets
admin@bastion:~$
```

create

passhport-admin target create [((<name> <hostname>) [–login=<login>] [–type=<ssh>] [–comment=<comment>] [–sshoptions=<sshoptions>] [–port=<port>])] creates a new target.

Argument	Description
<name>	Name of the target to create
hostname	Hostname or IP of the target
–login	Login to use when accessing the target (optional)
–type	The type of the target (for the commercial version only). It can be *ssh*, *postgresql*, *mysql*, *oracle*. This is used to know which hook to launch, depending on the server type. If type is something else than *ssh*, the server won't be accessible via SSH. If the target is a PostGreSQL server and you want to launch the corresponding hook (usually a proxy to log user actions, use *postgresql* type). Same explanations for *mysql* and *oracle*. Use the default *ssh*, unless you have the commercial version.
–comment	Comment concerning the target (optional)
–sshoptions	SSH options to use when connecting to the target (optional)
–port	SSH port to use when connecting to the target (optional)

Example :

```
admin@bastion:~# passhport-admin target create firewall.compagny.com 87.65.43.219 --
↪login=root --comment="Client 1 web server number 1"
OK: "firewall.compagny.com" -> created
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target create
Name: firewall2.compagny.com
Hostname: 87.65.43.220
Type (default is ssh):
Login (default is root):
Port: 22
SSH Options:
Comment: Client 1 FireWall 2 (Cisco)
OK: "firewall1.compagny.com" -> created
admin@bastion:~#
```

Once the target is created, you should add a passhport ssh public key to the target and use "checkaccess" to verify everything is ok.

edit

passhport-admin target edit [(<name> [–newname=<name>] [–newhostname=<hostname>] [–newtype=<ssh>] [–newlogin=<login>] [–newcomment=<comment>] [–newsshoptions=<sshoptions>] [–newport=<port>])] edits an existing target.

Argument	Description
<name>	Name of the target to edit
–newname	New name of the target if you want to rename it (optional)
–newhostname	New hostname/IP of the target (optional)
–newtype	The type of the target (for the commercial version only). It can be *ssh*, *postgresql*, *mysql*, *oracle*. This is used to know which hook to launch, depending on the server type. If type is something else than *ssh*, the server won't be accessible via SSH. If the target is a PostGreSQL server and you want to lauch the corresponding hook (usually a proxy to log user actions, use *postgresql* type). Same explanations for *mysql* and *oracle*. Use the default *ssh*, unless you have the commercial version.
–newlogin	New login to use when accessing the target (optional)
–newcomment	New comment concerning the target (optional)
–newsshoptions	New SSH options to use when connecting to the target (optional)
–newport	New SSH port to use when connecting to the target (optional)

Example :

```
admin@bastion:~# passhport-admin target edit firewall.compagny.com --
↪newname=firewall1.compagny.com --newcomment="Client 1 FireWall 1 (Cisco)" --
↪newlogin="admin"
OK: "firewall.compagny.com" -> edited
admin@bastion:~#
```

If no argument is given, user enters in interactive mode. It firsts shows all parameters of the target, then displays each parameters for a change. User can keep any previous configured parameter, just by typing "Enter". They only

exception is the comment. If user wants to remove the comment, he just type "Enter", and will then be asked if the original comment should be removed or not.

Example :

```
admin@bastion:~# passhport-admin target edit
Name of the target you want to modify: firewall2.compagny.com
Name: firewall2.compagny.com
Hostname: 87.65.43.220
Server Type : ssh
Login: root
Port: 22
SSH options:
Comment: Client 1 FireWall 2 (Cisco)
Attached users:
Usergroup list:
Users who can access this target:
All usergroups:
Member of the following targetgroups:
New name:
New hostname:
New Login: admin
New port:
New SSH options:
New comment:
Remove original comment? [y/N]N
OK: "firewall2.compagny.com" -> edited
admin@bastion:~#
```

As you can see above, we only changed the "New Login" entry. If an entry is simply replied with "enter", it keeps the previous value.

adduser

passhport-admin target adduser [(<username> <targetname>)] connects a target directly to a user.

Argument	Description
<username>	Name of the user to connect to the target
<targetname>	Name of the target on which to connect the user

Example :

```
admin@bastion:~# passhport-admin target adduser admin1@compagny.com firewall1.
↪compagny.com
OK: "admin1@compagny.com" added to "firewall1.compagny.com"
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target adduser
Username: admin2@compagny.com
Targetname: firewall2.compagny.com
OK: "admin2@compagny.com" added to "firewall2.compagny.com"
admin@bastion:~#
```

rmuser

passhport-admin target rmuser [(<username> <targetname>)] deletes the direct connection between a target and a user.

Argument	Description
<username>	Name of the user to disconnect to the target
<targetname>	Name of the target on which to disconnect the user

Example :

```
admin@bastion:~# passhport-admin target rmuser admin1@compagny.com firewall1.compagny.
↪com
OK: "admin1@compagny.com" removed from "firewall1.compagny.com"
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target rmuser
Username: admin2@compagny.com
Targetname: firewall2.compagny.com
OK: "admin2@compagny.com" removed from "firewall2.compagny.com"
admin@bastion:~#
```

addusergroup

passhport-admin target addusergroup [(<usergroupname> <targetname>)] connects a target directly to a usergroup.

Argument	Description
<usergroupname>	Name of the usergroup to connect to the target
<targetname>	Name of the target on which to connect the usergroup

Example :

```
admin@bastion:~# passhport-admin target addusergroup firewall-admins firewall1.
↪compagny.com
OK: "firewall-admins" added to "firewall1.compagny.com"
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target addusergroup
Usergroupname: firewall-admins
Targetname: firewall2.compagny.com
OK: "firewall-admins" added to "firewall2.compagny.com"
admin@bastion:~#
```

rmusergroup

passhport-admin target delusergroup [(<usergroupname> <targetname>)] delete the connection between a target and a usergroup.

Argument	Description
<usergroupname>	Name of the usergroup to disconnect to the target
<targetname>	Name of the target on which to disconnect the usergroup

Example :

```
admin@bastion:~# passhport-admin target addusergroup firewall-admins firewall1.
↪compagny.com
OK: "firewall-admins" added to "firewall1.compagny.com"
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target addusergroup
Usergroupname: firewall-admins
Targetname: firewall2.compagny.com
OK: "firewall-admins" added to "firewall2.compagny.com"
admin@bastion:~#
```

delete

passhport-admin target delete [([-f | –force] <name>)] delete a target.

Argument	Description
<name>	Name of the target to delete
-f or –force	If used, user won't be prompt for confirmation

Example :

```
admin@bastion:~# passhport-admin target delete firewall1.compagny.com
Name: firewall1.compagny.com
Hostname: firewall1.compagny.com
Server Type : ssh
Login: admin
Port: 22
SSH options:
Comment: Client 1 FireWall 1 (Cisco)
Attached users:
Usergroup list: firewall-admins
Users who can access this target:
All usergroups: firewall-admins
Member of the following targetgroups:
Are you sure you want to delete firewall1.compagny.com? [y/N] y
OK: "firewall1.compagny.com" -> deleted
admin@bastion:~#
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~# passhport-admin target delete
Name: firewall2.compagny.com
Name: firewall2.compagny.com
Hostname: 87.65.43.220
Server Type : ssh
Login: admin
Port: 22
SSH options:
Comment: Client 1 FireWall 2 (Cisco)
Attached users:
Usergroup list: firewall-admins network-admins
Users who can access this target:
All usergroups: firewall-admins network-admins
Member of the following targetgroups:
Are you sure you want to delete firewall2.compagny.com? [y/N] y
OK: "firewall2.compagny.com" -> deleted
admin@bastion:~#
```

1.4.4 usergroup

Usages :

```
passhport-admin usergroup list
passhport-admin usergroup search [<pattern>]
passhport-admin usergroup show [<name>]
passhport-admin usergroup create [(<name> [--comment=<comment>])]
passhport-admin usergroup edit [(<name> [--newname=<name>] [--newcomment=<comment>])]
passhport-admin usergroup (adduser | rmuser) [(<username> <usergroupname>)]
passhport-admin usergroup (addusergroup | rmusergroup) [(<subusergroupname>
↪<usergroupname>)]
passhport-admin usergroup delete [([-f | --force] <name>)]
```

list

passhport-admin usergroup list show all the configured usergroups.

Example :

```
admin@bastion:~$ passhport-admin usergroup list
admins
database-admins
external
network-admins
admin@bastion:~$
```

search

passhport-admin usergroup search [<PATTERN>] searches in the usergroup list for all usergroups that correspond to the given pattern.

Example :

```
admin@bastion:~$ passhport-admin usergroup search ext
external
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup search
Pattern: admins
admins
database-admins
network-admins
admin@bastion:~$
```

show

passhport-admin usergroup show <*NAME*> shows informations about the usergroup <NAME>.

Example :

```
admin@bastion:~$ passhport-admin usergroup show admins
Name: admins
Comment:
User list: john@compagny.com vincent@compagny.com
Usergroup list:
All users: john@compagny.com vincent@compagny.com
All usergroups:
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup show
Name: admins
Name: admins
Comment:
User list: john@compagny.com vincent@compagny.com
Usergroup list:
All users: john@compagny.com vincent@compagny.com
All usergroups:
admin@bastion:~$
```

create

passhport-admin usergroup create [(<name> [–comment=<comment>])] creates a new usergroup.

Argument	Description
<name>	Name of the usergroup to create
–comment	Comment concerning the usergroup (optional)

Example :

```
admin@bastion:~$ passhport-admin usergroup create external
OK: "external" -> created
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup delete database-admins
Name: database-admins
Comment:
User list:
Usergroup list:
All users:
All usergroups:
Are you sure you want to delete database-admins? [y/N] y
OK: "database-admins" -> deleted
admin@bastion:~$
```

edit

passhport-admin usergroup edit [(<name> [–newname=<name>] [–newcomment=<comment>])] edits an existing usergroup.

Argument	Description
<name>	Name of the usergroup to edit
–newname	New name of the usergroup if you want to rename it (optional)
–newcomment	New comment concerning the usergroup (optional)

Example :

```
admin@bastion:~$ passhport-admin usergroup edit admins --newname=linux-admins
OK: "admins" -> edited
admin@bastion:~$
```

If no argument is given, user enters in interactive mode. It firsts shows all parameters of the usergroup, then displays each parameters for a change. User can keep any previous configured parameter, just by typing "Enter". They only exception is the comment. If user wants to remove the comment, he just type "Enter", and will then be asked if the original comment should be removed or not.

Example :

```
admin@bastion:~$ passhport-admin usergroup edit
Name of the usergroup you want to modify: external
Name: external
Comment:
User list:
Usergroup list:
All users:
All usergroups:
New name: external-admins
New comment:
Remove original comment? [y/N]
OK: "external" -> edited
admin@bastion:~$
```

As you can see above, we only changed the "New name" entry. If an entry is simply replied with "enter", it keeps the previous value.

adduser

passhport-admin usergroup adduser [(<username> <usergroupname>)] add a user in a usergroup.

Argument	Description
<username>	Name of the user to add in a usergroup
<usergroupname>	Name of the usergroup in which to add the user

Example :

```
admin@bastion:~$ passhport-admin usergroup adduser vincent@compagny.com network-admins
OK: "vincent@compagny.com" added to "network-admins"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup adduser
Username: yann@ext-compagny.com
Usergroupname: external-admins
OK: "yann@ext-compagny.com" added to "external-admins"
admin@bastion:~$
```

rmuser

passhport-admin usergroup rmuser [(<username> <usergroupname>)] removes a user from a usergroup.

Argument	Description
<username>	Name of the user to remove from a usergroup
<usergroupname>	Name of the usergroup of which to remove the user

Example :

```
admin@bastion:~$ passhport-admin usergroup rmuser vincent@compagny.com linux-admins
OK: "vincent@compagny.com" removed from "linux-admins"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup rmuser
Username: yann@ext-compagny.com
Usergroupname: external-admins
OK: "yann@ext-compagny.com" removed from "external-admins"
admin@bastion:~$
```

addusergroup

passhport-admin usergroup addusergroup [(<subusergroupname> <usergroupname>)] adds a usergroup in another usergroup.

Argument	Description
<subusergroupname>	Name of the usergroup to add in a usergroup
<usergroupname>	Name of the usergroup in which to add the usergroup

Example :

```
admin@bastion:~$ passhport-admin usergroup addusergroup linux-admins admins
OK: "linux-admins" added to "admins"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup addusergroup
Subusergroupname: network-admins
Usergroupname: admins
OK: "network-admins" added to "admins"
admin@bastion:~$
```

rmusergroup

passhport-admin usergroup rmusergroup [(<subusergroupname> <usergroupname>)] delete the connection between a usergroup and a usergroup.

Argument	Description
<subusergroupname>	Name of the usergroup to remove from a usergroup
<usergroupname>	Name of the usergroup of which to remove the usergroup

Example :

```
admin@bastion:~$ passhport-admin usergroup rmusergroup linux-admins admins
OK: "linux-admins" removed from "admins"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup rmusergroup
Subsergroupname: network-admins
Usergroupname: admins
OK: "network-admins" removed from "admins"
admin@bastion:~$
```

delete

passhport-admin usergroup delete [([-f | –force] <name>)] delete a usergroup.

Argument	Description
<name>	Name of the usergroup to delete
-f or –force	If used, user won't be prompt for confirmation

Example :

```
admin@bastion:~$ passhport-admin usergroup delete network-admins
Name: network-admins
Comment:
User list: vincent@compagny.com
Usergroup list:
All users: vincent@compagny.com
All usergroups:
Are you sure you want to delete network-admins? [y/N] y
OK: "network-admins" -> deleted
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin usergroup delete
Name: linux-admins
Name: linux-admins
Comment:
User list: john@compagny.com
Usergroup list:
All users: john@compagny.com
All usergroups:
Are you sure you want to delete linux-admins? [y/N] y
OK: "network-admins" -> deleted
admin@bastion:~$
```

1.4.5 targetgroup

Usages :

```
passhport-admin targetgroup list
passhport-admin targetgroup search [<pattern>]
passhport-admin targetgroup show [<name>]
passhport-admin targetgroup create [(<name> [--comment=<comment>])]
passhport-admin targetgroup edit [(<name> [--newname=<name>] [--newcomment=<comment>
↪])]
passhport-admin targetgroup (adduser | rmuser) [(<username> <targetgroupname>)]
passhport-admin targetgroup (addtarget | rmtarget) [(<targetname> <targetgroupname>)]
passhport-admin targetgroup (addusergroup | rmusergroup) [(<usergroupname>
↪<targetgroupname>)]
passhport-admin targetgroup (addtargetgroup | rmtargetgroup) [(<subtargetgroupname>
↪<targetgroupname>)]
passhport-admin targetgroup delete [([-f | --force] <name>)]
```

list

passhport-admin targetgroup list show all the configured targetgroups.

Example :

```
admin@bastion:~$ passhport-admin targetgroup list
linux-servers
network-appliances
phone-appliance
admin@bastion:~$
```

search

passhport-admin targetgroup search [<PATTERN>] searches in the targetgroup list for all targetgroups that correspond to the given pattern.

Example :

```
admin@bastion:~$ passhport-admin targetgroup search appliance
network-appliances
phone-appliance
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup search
Pattern: servers
linux-servers
admin@bastion:~$
```

show

passhport-admin targetgroup show <*NAME*> shows informations about the targetgroup <NAME>.

Example :

```
admin@bastion:~$ passhport-admin targetgroup show linux-servers
Name: linux-servers
Comment:
User list:
Target list: linux-7892 linux-7239 linux-1398
Usergroup list:
Targetgroup list:
All users:
All targets: linux-1398 linux-7239 linux-7892
All usergroups:
All targetgroups:
admin@bastion:~$
```

If no pattern is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup show
Name: linux-servers
Name: linux-servers
Comment:
```

(continues on next page)

(continued from previous page)

```
User list:
Target list: linux-7892 linux-7239 linux-1398
Usergroup list:
Targetgroup list:
All users:
All targets: linux-1398 linux-7239 linux-7892
All usergroups:
All targetgroups:
admin@bastion:~$
```

create

passhport-admin targetgroup create [(<name> [–comment=<comment>])] creates a new targetgroup.

Argument	Description
<name>	Name of the targetgroup to create
–comment	Comment concerning the targetgroup (optional)

Example :

```
admin@bastion:~$ passhport-admin targetgroup create linux-servers
OK: "linux-servers" -> created
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup create
Name: phone-appliance
Comment: Phones and IPBX appliances group.
OK: "phone-appliance" -> created
admin@bastion:~$
```

edit

passhport-admin targetgroup edit [(<name> [–newname=<name>] [–newcomment=<comment>])] edits an existing targetgroup.

Argument	Description
<name>	Name of the targetgroup to edit
–newname	New name of the targetgroup (optional)
–newcomment	New comment concerning the targetgroup (optional)

Example :

```
admin@bastion:~$ passhport-admin targetgroup edit linux-servers --newcomment="Linux␣
↪servers group."
OK: "linux-servers" -> edited
admin@bastion:~$
```

If no argument is given, user enters in interactive mode. It firsts shows all parameters of the target, then displays each parameters for a change. User can keep any previous configured parameter, just by typing "Enter". They only exception is the comment. If user wants to remove the comment, he just type "Enter", and will then be asked if the original comment should be removed or not.

Example :

```
admin@bastion:~$ passhport-admin targetgroup edit
Name of the targetgroup you want to modify: network-appliances
Name: network-appliances
Comment:
User list:
Target list:
Usergroup list:
Targetgroup list:
All users:
All targets:
All usergroups:
All targetgroups:
New name:
New comment: Network appliance group.
OK: "network-appliances" -> edited
admin@bastion:~$
```

As you can see above, we only changed the "New comment" entry. If an entry is simply replied with "enter", it keeps the previous value.

adduser

passhport-admin targetgroup adduser [(<username> <targetgroupname>)] connects a targetgroup directly to a user.

Argument	Description
<username>	Name of the user to connect to the targetgroup
<targetname>	Name of the targetgroup on which to connect the user

Example :

```
admin@bastion:~$ passhport-admin targetgroup adduser john@compagny.com linux-servers
OK: "john@compagny.com" added to "linux-servers"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup adduser
Username: vincent@compagny.com
Targetgroupname: network-appliances
OK: "vincent@compagny.com" added to "network-appliances"
admin@bastion:~$
```

rmuser

passhport-admin targetgroup rmuser [(<username> <targetgroupname>)] deletes the direct connection between a targetgroup and a user.

Argument	Description
<username>	Name of the user to disconnect to the targetgroup
<targetname>	Name of the targetgroup of which to disconnect the user

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmuser vincent@compagny.com network-
↪appliances
OK: "vincent@compagny.com" removed from "network-appliances"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmuser
Username: john@compagny.com
Targetgroupname: linux-servers
OK: "john@compagny.com" removed from "linux-servers"
admin@bastion:~$
```

addusergroup

passhport-admin targetgroup addusergroup [(<usergroupname> <targetgroupname>)] connects a targetgroup directly to a usergroup.

Argument	Description
<usergroupname>	Name of the usergroup to connect to a targetgroup
<targetname>	Name of the targetgroup on which to connect the usergroup

Example :

```
admin@bastion:~$ passhport-admin targetgroup addusergroup linux-admins linux-servers
OK: "linux-admins" added to "linux-servers"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup addusergroup
Usergroupname: network-admins
Targetgroupname: network-appliances
OK: "network-admins" added to "network-appliances"
admin@bastion:~$
```

rmusergroup

passhport-admin targetgroup delusergroup [(<usergroupname> <targetgroupname>)] delete the connection between a targetgroup and a usergroup.

Argument	Description
<usergroupname>	Name of the usergroup to disconnect to the targetgroup
<targetname>	Name of the targetgroup of which to disconnect the usergroup

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmusergroup linux-admins linux-servers
OK: "linux-admins" removed from "linux-servers"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmusergroup
Usergroupname: network-admins
Targetgroupname: network-appliances
OK: "network-admins" removed from "network-appliances"
admin@bastion:~$
```

addtargetgroup

passhport-admin targetgroup addtargetgroup [(<subtargetgroupname> <targetgroupname>)] connects a subtargetgroup directly to a targetgroup.

Argument	Description
<subtargetgroupname>	Name of the subtargetgroup to connect to a targetgroup
<targetname>	Name of the targetgroup with which to connect the subtargetgroup

Example :

```
admin@bastion:~$ passhport-admin targetgroup addtargetgroup linux-servers all-servers
OK: "linux-servers" added to "all-servers"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup addtargetgroup
Subtargetgroupname: network-appliances
Targetgroupname: all-servers
OK: "network-appliances" added to "all-servers"
admin@bastion:~$
```

rmtargetgroup

passhport-admin targetgroup deltargetgroup [(<subtargetgroupname> <targetgroupname>)] delete the connection between a subtargetgroup and a targetgroup.

Argument	Description
<subtargetgroupname>	Name of the subtargetgroup to disconnect to a targetgroup
<targetname>	Name of the targetgroup of which to disconnect the subtargetgroup

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmtargetgroup linux-servers all-servers
OK: "linux-servers" removed from "all-servers"
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup rmtargetgroup
Subtargetgroupname: network-appliances
Targetgroupname: all-servers
OK: "network-appliances" removed from "all-servers"
admin@bastion:~$
```

delete

passhport-admin targetgroup delete [([-f | –force] <name>)] delete a target.

Argument	Description
<name>	Name of the targetgroup to delete
-f or –force	If used, user won't be prompt for confirmation

Example :

```
admin@bastion:~$ passhport-admin targetgroup delete linux-servers
Name: linux-servers
Comment: Linux servers group.
User list:
Target list: linux-7892 linux-7239 linux-1398
Usergroup list:
Targetgroup list:
All users:
All targets: linux-1398 linux-7239 linux-7892
All usergroups:
All targetgroups:
Are you sure you want to delete linux-servers? [y/N] y
OK: "linux-servers" -> deleted
admin@bastion:~$
```

If no argument is given, user enters in interactive mode.

Example :

```
admin@bastion:~$ passhport-admin targetgroup delete
Name: network-appliances
Name: network-appliances
Comment: Network appliance group.
User list:
Target list:
Usergroup list:
Targetgroup list:
All users:
All targets:
All usergroups:
All targetgroups:
```

(continues on next page)

(continued from previous page)

```
Are you sure you want to delete network-appliances? [y/N] y
OK: "linux-servers" -> deleted
admin@bastion:~$
```

1.5 User-side usage

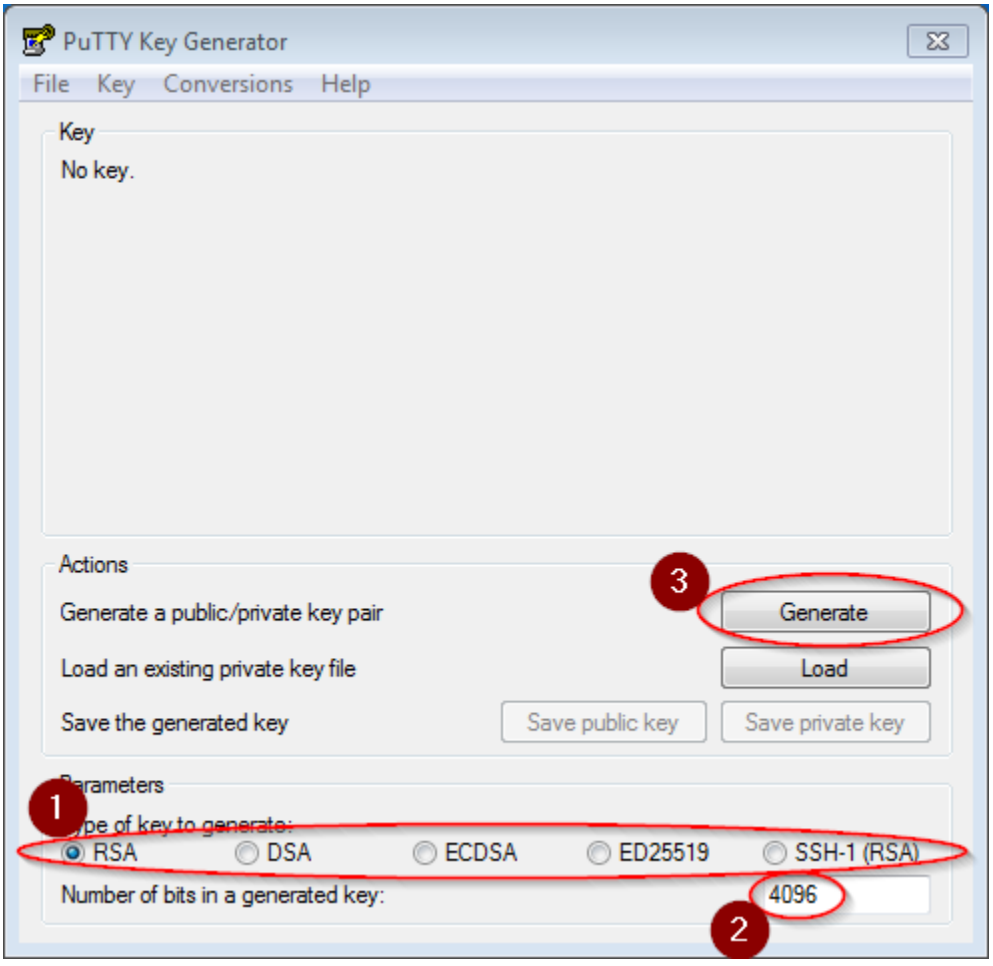
This chapter shows how to connect to a target, through PaSSHport, from key generation, to SCP.

1.5.1 Generate private keys

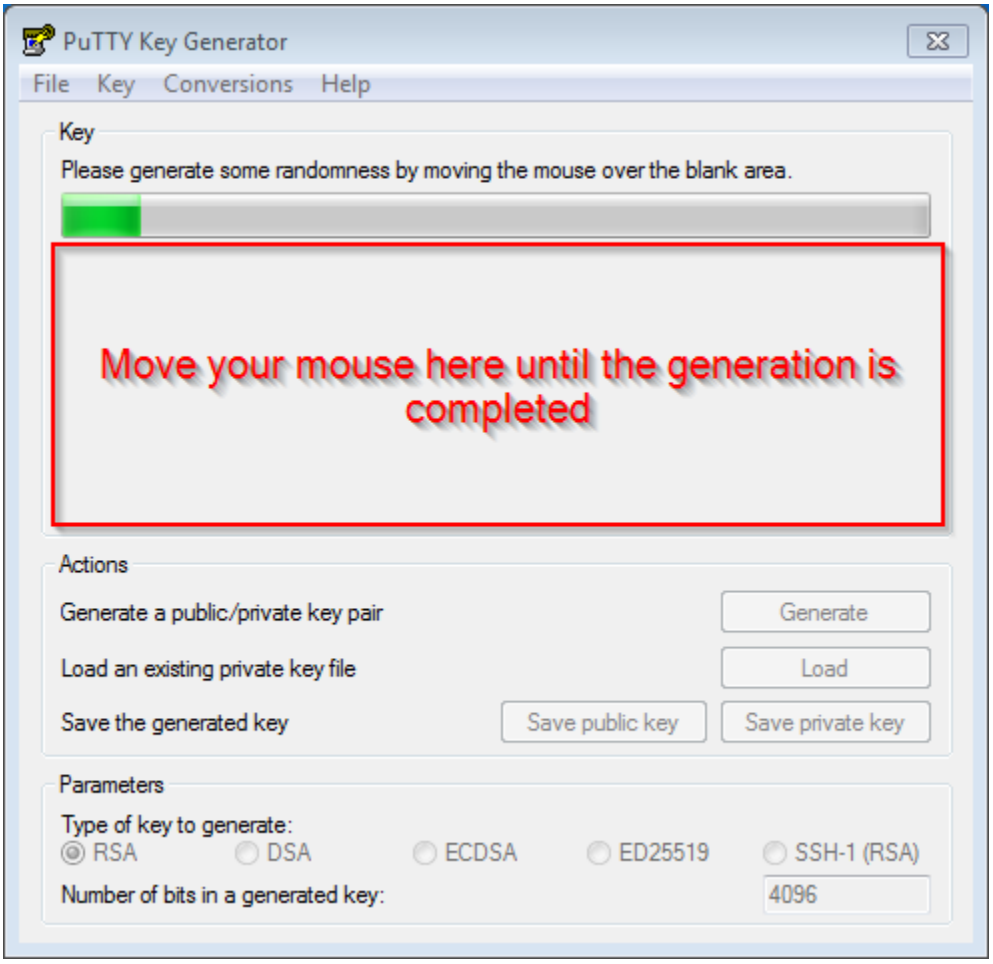
On Windows :

To generate the public key (extract from the along side generated private key) that you'll give to your PaSSHport admin, use *puttygen* that you can download from here (search for *puttygen.exe*).

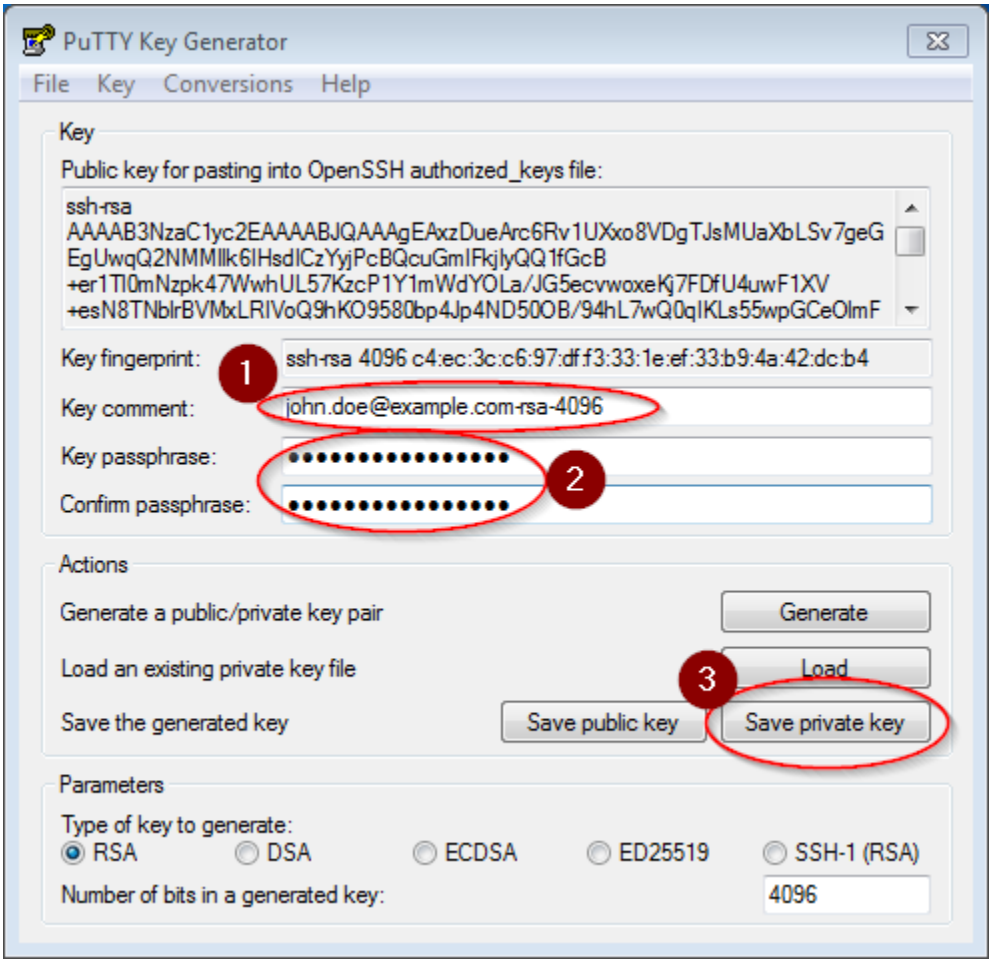
Start puttygen, and on the main windows, select the type of key you want to generate (*1*), the key length (*2*), then click *Generate* button (*3*). Here, we selected RSA and a key length of 4096 (2048 is a considered as a minimum for RSA) :



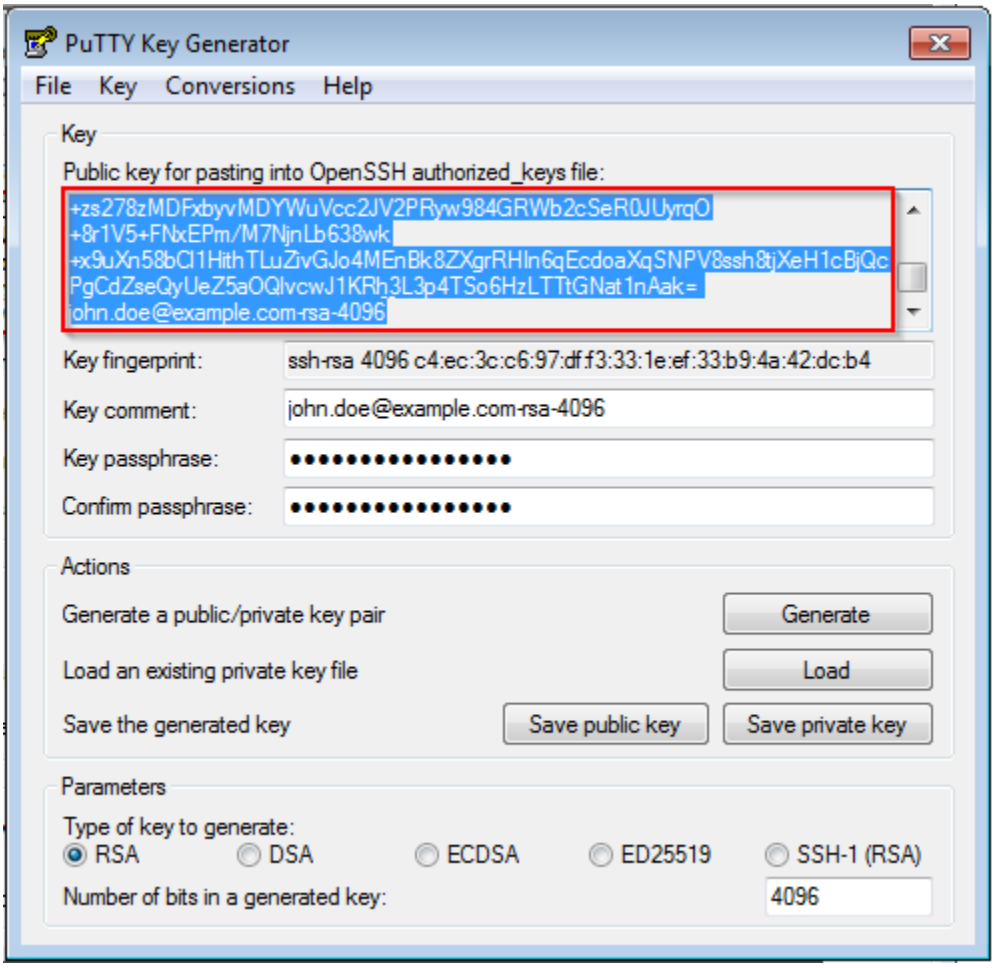
Move your mouse in the blank space, until the key is generated :



Once generated, insert a comment (*1*), a strong passphrase (*2*), then save your private key (*3*).



You now have to send your RSA public key to your PaSSHport admin. Select your public key as shown in this screen capture, and copy/paste it into a mail to your PaSSHport admin :



You now have to wait until your PaSSHport admin add your key to your account into PaSSHport.

On Linux / Unix :

Simply open a terminal, and use the *ssh-keygen* command. Here we generate à 4096bits length RSA key :

```
user@host:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1r28XcYMIclivAHSSqmzH5Dh1LJ+IMsQMhl2Ds1HtXQ user@passhport-debian9-dev
The key's randomart image is:
+---[RSA 4096]----+
|.=o..oo=.E       |
|* +o+.=.+o . .   |
|.o +.B o  = + .  |
|. . O .  o = . . |
| o + =  S o . .  |
```

(continues on next page)

(continued from previous page)

```
|  o o o.    . . + |
|     o .     o   =|
|       .       o o |
|             . .   |
+----[SHA256]-----+
user@host:~$
```

Display your newly created public key :

```
user@host:~$ cat .ssh/id_rsa.pub
ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAACAQDmcmVG6uGW3BvOkHN7M7ubITihVwL9glc7jilZvzgDJL4CzCXG2VwjdxHaCBfW82HsgoU
↪TEZw/tgUWSSo7TOmldFmEs4TmZc9n0lhCgGT/XtShqtwyYAxeAw419Uc+L/
↪unXKPRtulLjNqdp62GW68CTQ7GzJosDWLYWZfNrhRoMvw6K6j/
↪vLbVcoktY+RNoNdFjYhgPcKzP0p73pvlh9uIKohBkh3vh5pOfVEu6L9J4VvjM3dACScPJORG05N7MB4rJ3FpSy9fgfMwaT99Xm7
↪IVKXZxoUjB9z2EBkKYK+Hlj5Oopwgas6AvcrJIdZo1tsbdUYqcbQKoX7TeSwjbxESygFuCLMgs4SuMy8/
↪1+pPIlJQY7XzdCCDzkEp/
↪s12Ca5xPSUpFGdWKKIf1jzZzjS5BeUzm63ldFoN+HHKuU7FRpPNSXrlWNkqkwHnpa1SbhT3yOlu6BdnxMcaRNAeQ+cfxyykUSdO
↪toXNiXpyhrg3RT35Pj96cx7nwg9CrQ== user@passhport-debian9-dev
user@host:~$
```

And send this content to your PaSSHport admin. You now have to wait until your PaSSHport admin add your key to your account into PaSSHport.

1.5.2 Connect to PaSSHport

On Windows

You can use Putty to connect to the passhport bastion indicating your private key.

On Linux / *nix

Use your standard SSH client and connect to the bastion as passhport user:

```
ssh passhport@bastion.tld
```

If you want to be directly connected, you have to specify the target name in the command AND to activate the -t option:

```
ssh -t passhport@bastion.tld targetname
```

If you want to launch a command directly, you can use the same syntax adding the command at the end

```
ssh -t passhport@bastion.tld targetname 'cat /proc/cpuinfo'
```

1.5.3 scp through PaSSHport

Using CLI

To use SCP throug PaSSHport, use one of the following syntax...

From target to local :

```
$ scp passhport@my-passhport-server:TARGET_NAME//etc/fstab /tmp/.
```

From local to target :

```
$ scp /etc/passwd passhport@my-passhport-server:TARGET_NAME//tmp/.
```

For Windows, you have to consider using "pscp" command. You can find it here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html Be aware that you will need a private key file in "ppk" format. You can use puttygen to generate one or convert your current private key. Pscp command tries to use SFTP protocol leading to this error:

```
FATAL ERROR: Received unexpected end-of-file from server
```

In order to use pscp with PaSSHport, consider using this syntax including "-scp" flag:

```
pscp -scp -i "C:\path\to\sshkeys\my_private_key.ppk"  "C:\path\to\file\totransfer.txt
↪" passhport@my-passhport-server:TARGET_NAME//pah/to/copy
```

Some links :

- Project Site : <http://www.passhport.org>
- Github : <https://www.github.com/LibrIT/passhport>