OpenPGPkeyserver

Building a PGP SKS Keyserver

Notice! This page is written with Ubuntu 16.04 LTS in mind, please see the Ubuntu 14.04 LTS (/guides/building-server/ubuntu-14/) version of this document.

A Key Server is used to distribute PGP/GPG (http://en.wikipedia.org/wiki/Pretty_Good_Privacy) keys between different users. One of the most popular key servers for use with pgp/gpg is the sks keyserver (https://bitbucket.org/skskeyserver/sks-keyserver). This document will walk you through downloading, installing, and setting up a sks keyserver on Ubuntu (http://www.ubuntu.com/) 16.04 LTS.

Building your own PGP SKS Server

Building a SKS (https://bitbucket.org/skskeyserver/sks-keyserver) server is a pretty straight forward project if you are use to running servers.

To build a production SKS Server, you must…

- Building the SKS Daemon
- Download the needed database files
- Import the downloaded databases files
- Configure your web-server
- Start the SKS Daemon
- Configure your web-server
- Install the SKS webpage on your server
- Start the SKS Daemon
- Patches for the sks-keyserver software

Building the SKS Daemon

The following is for Ubuntu (http://www.ubuntu.com/) 16.04 LTS

```
apt-get -y install gcc ocaml libdb-dev gnupg nginx wget
```

After installing the required software, you need to download SKS

```
gpg --keyserver hkp://pool.sks-keyservers.net --trust-model always --recv-key 0x0B7F8B60E3EDFAE3
wget https://bitbucket.org/skskeyserver/sks-keyserver/downloads/sks-1.1.6.tgz
wget  https://bitbucket.org/skskeyserver/sks-keyserver/downloads/sks-1.1.6.tgz.asc
gpg --keyid-format long --verify sks-1.1.6.tgz.asc
```

The output of the last command should be

```
gpg: Signature made Mon 05 May 2014 02:06:51 PM CDT
gpg:                using RSA key 41259773973A612A
gpg: Good signature from "SKS Keyserver Signing Key"
```

Now, untar the software

```
tar -xzf sks-1.1.6.tgz
cd sks-1.1.6
```

Next copy the **Makefile.local.unused** to **Makefile.local** and change `ldb-4.6 to ldb-5.3` for Ubuntu.

```
cp Makefile.local.unused Makefile.local
sed -i 's/ldb\-4.6/ldb\-5.3/' Makefile.local
```

Last, build the software

```
make dep
make all
make install
```

Configure the sks-keyserver

/var/lib/sks/sksconf

```
# /var/lib/sks/sksconf

debuglevel: 3

# Set the hostname of your server
hostname:                       --keyserver-hostname--

hkp_address:                    127.0.0.1
hkp_port:                       11371
recon_port:                     11370

# Set the PGP ID for the Server Contact
server_contact:                 --contact-pgp-id--

initial_stat:
disable_mailsync:
membership_reload_interval:     1
stat_hour:                      12

max_matches:                    500
```

Download the needed database files

Rather than starting with an empty database and attempting to populate it by syncing with other keyservers (a bad idea because it loads up your peers with lots of traffic and will probably fail anyway with deadlocks in the conflict resolution system) we'll grab a static dump from an existing SKS server. Currently the only known source is:

- http://keyserver.mattrude.com/dump/ (http://keyserver.mattrude.com/dump/) - Generated every DAY
- ftp://ftp.prato.linux.it/pub/keyring/ (ftp://ftp.prato.linux.it/pub/keyring/) - Generated every Wednesday
- http://keyserver.borgnet.us/dump (http://keyserver.borgnet.us/dump) - Generated every Sunday

Download Keydump

The keydump is about 7.3GB as of Oct 2015, increasing at a rate of about one gigabyte per year, so fetching it will take a long time. It's divided into a bunch of individual numbered files so you'll need to fetch all of them. Because I'm too lazy to spend 8 hours sitting there doing it manually I did it like this:

```
mkdir -p /var/lib/sks/dump
cd /var/lib/sks/dump
wget -crp -e robots=off --level=1 --cut-dirs=3 -nH \
-A pgp,txt https://keyserver.mattrude.com/dump/current/
```

Many hours later, check that all the pieces downloaded correctly by comparing their checksums against the list published by the dump provider:

```
md5sum -c metadata-sks-dump.txt
```

Import the downloaded databases files

There are two ways to do this: either a full build (which reads in the dump you just downloaded and leaves you with a complete, self-contained database) or a fastbuild (which just references the dump and requires it to be left in place after the fastbuild is complete). I started doing a full build, it looked like it was going to take forever so I aborted it and switched to a quickbuild. On the 4-processor machine I was using it still took in the order of 40 minutes to run so this might take a while.

You need to be in the basedir when running this and the dumps have to be in a sub-directory called `dump` (which they should be if you followed the steps above), so:

```
cd /var/lib/sks
```

Then run the build

```
/usr/local/bin/sks_build.sh
```

On the next screen, choose **2**.

```
Please select the mode in which you want to import the keydump:

1 - fastbuild
    only an index of the keydump is created and the keydump cannot be
    removed.

2 - normalbuild

    all the keydump will be imported in a new database. It takes longer
    time and more disk space, but the server will run faster (depending
    from the source/age of the keydump).
    The keydump can be removed after the import.

Enter enter the mode (1/2): 2
```

If you edit the `sks_build.sh` script you'll discover it's just a shell script which calls SKS itself to do the heavy lifting. If you have trouble with lack of memory you may need to tweak the script a bit: in particular the `-n 10` flag used in the fastbuild call is a multiple of 15,000 keys to load at a time. The default setting therefore loads 150,000 keys at a time which could cause your machine to go into swap, and changing to something like `-n 2` will cause it to load only 30,000 at a time instead and possibly complete the job faster. The trick is to load as many as possible in each pass without hitting swap - if that happens, performance falls through the floor and you may as well abort it and start again (after deleting the KDB and PTree directories created by the aborted import).

If all goes smoothly you'll end up with `KDB` and `PTree` directories in `/var/lib/sks`.

Configure your web-server

Inorder to be part of the sks pool (https://sks-keyservers.net/status/), among other things, you need to route your sks traffic threw a web proxy. If you followed these instructions you should have already installed the nginx software on your server, all you should have to do is copy the below config into the `/etc/nginx/nginx.conf` file on your server and change the IP addresses to meeting your setup.

You will need to change `--IPv4-Address--` to be your IPv4 IP address, and if you have one, change `--IPv6-Address--` to be your IPv6 address. If you do not have a IPv6 address on your server, you should remove that line from the configuraton.

/etc/nginx/nginx.conf

```
#/etc/nginx/nginx.conf

user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
    worker_connections 768;
}

http {
    sendfile    on;
    tcp_nopush  on;
    tcp_nodelay on;
    client_max_body_size 8m;

    access_log  /var/log/nginx/access.log;
    error_log   /var/log/nginx/error.log;
    rewrite_log on;

    include /etc/nginx/mime.types;

    #----------------------------------------------------------------------
    # OpenPGP Public SKS Key Server
    #----------------------------------------------------------------------

    server {
        listen 80 default_server;
        listen [::]:80 default_server;
        listen --IPv4-Address--:11371 default_server;
        listen [--IPv6-Address--]:11371 default_server;

        server_name *.sks-keyservers.net;
        server_name *.pool.sks-keyservers.net;
        server_name pgp.mit.edu;
        server_name keys.gnupg.net;

        root /var/www/html;

        rewrite ^/stats /pks/lookup?op=stats;
        rewrite ^/s/(.*) /pks/lookup?search=$1;
        rewrite ^/search/(.*) /pks/lookup?search=$1;
        rewrite ^/g/(.*) /pks/lookup?op=get&search=$1;
        rewrite ^/get/(.*) /pks/lookup?op=get&search=$1;
        rewrite ^/d/(.*) /pks/lookup?op=get&options=mr&search=$1;
        rewrite ^/download/(.*) /pks/lookup?op=get&options=mr&search=$1;

        location /pks {
            proxy_pass         http://127.0.0.1:11371;
            proxy_pass_header  Server;
            add_header         Via "1.1 --keyserver-hostname--:11371 (nginx)";
            proxy_ignore_client_abort on;
            client_max_body_size 8m;
        }
    }
}
```

Once you have copied this file into `/etc/nginx/nginx.conf` , you need to restart nginx by running the following command.

```
service nginx restart
```

Install the SKS webpage on your server

Now we need to install a webpage so visitors are able to interact with your new keyserver via their web browser. The sks-keyserver project has 3 older sites that come with the default install, these sites may be found in the source (https://bitbucket.org/skskeyserver/sks-keyserver/src/40280f59d0f503da1326972757168aa42335573f/sampleWeb/?at=default) on bitbucket.

There are obviously meny options besides the three provided by sks-keyserver. Were going to install pgpkeyserver-lite (https://github.com/mattrude/pgpkeyserver-lite), a example of this site may be found on keys.therudes.com (http://keys.therudes.com/). The install process is pretty straightforward, we will download the tarball, extract it, drop into the html directory we setup above (`/var/www/html`), and lastly update the infromation on the site to reflect your setup.

Download & extract the tarball

```
curl -Ls https://github.com/mattrude/pgpkeyserver-lite/tarball/master -o pgpkeyserver-lite.tgz &&
\
mkdir /var/www/html && tar -xzf pgpkeyserver-lite.tgz --directory /var/www/html --strip 1
```

Modify the site

After downloading and extracting the tarball, you need to modify the site to reflect the setup of your keyserver. There are two sections that need to be replaced. first you need to replace all instances of `###ENTERNAMEHERE###` with your own name. Next, replace all instances of `###ENTERPUBLICKEYHERE###` with your public key. Or you may of course modify the site in anyway you wish.

Start the SKS Daemon

/etc/init.d/sks