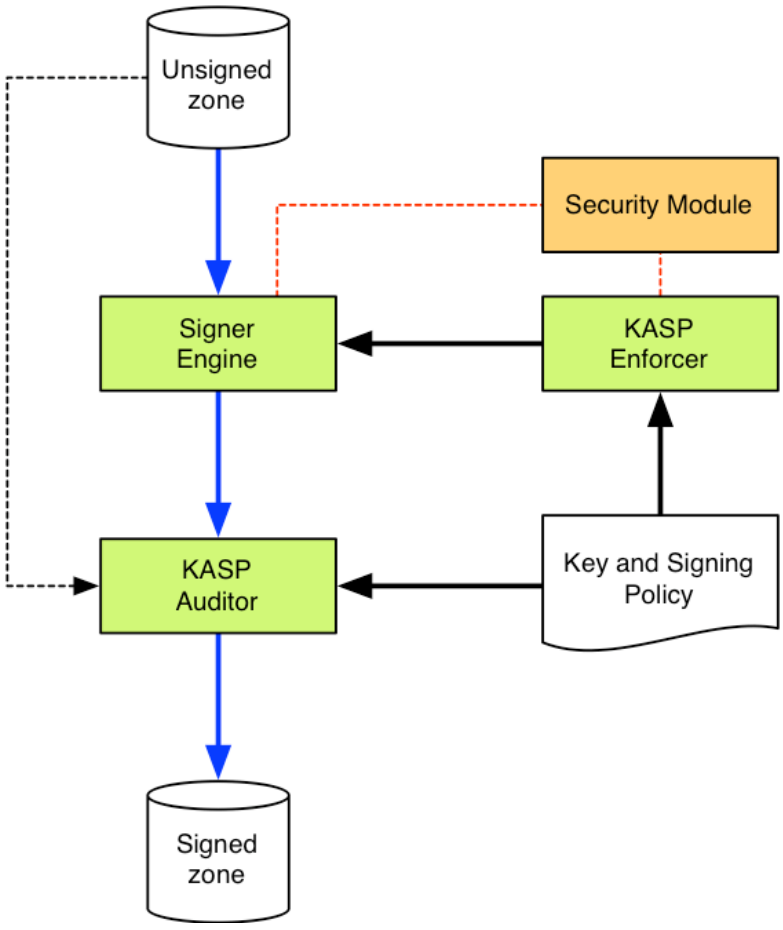
Using OpenDNSSEC

Scope

The goal of OpenDNSSEC is to have a complete DNSSEC zone signing system which is very easy to use with stability and security in mind. There are a lot of details in signing zone files with DNSSEC and OpenDNSSEC covers most of it. The scope if this document is to provide the user with enough documentation in order to get the system up and running with a basic configuration. More details about configuring all the parameters in the system can be found on the OpenDNSSEC wiki system, links are provided below at each section.

Overview

OpenDNSSEC is divided in a number of smaller components. A short overview gives you the Key and Signature Policy Enforcer (KASP Enforcer) which maintains a policy for the system, the Signer which do the actual signing, a keystore which is typically a Hardware Security Module (HSM) and the Auditor which does checks on the signed zone files to verify that the signing conforms to the policy given in KASP.



Installation

OpenDNSSEC depends on a number of open-source packages, all of which must be installed on your system for OpenDNSSEC to build successfully. This guide describes only the installation of the OpenDNSSEC software itself, refer to our wiki for an installation guide of the pre-requisites.

Pre-requisites

You need the following software in order to install and build OpenDNSSEC:

- **ldns** 1.6.0 or later.
- **libxml2**
- **botan** 1.8.0 or later
- **ruby**, **rubygems**, **dnsruby** and **openssl** for ruby (for the auditor)
- **trang** (only for building)
- **sqlite3**
- the Python XML library, **python-4suite-xml** or source from 4suite.org

Obtaining the Source Code

The development (unstable) version of OpenDNSSEC is available from the Subversion repository and can be obtained using the following command:

```
svn co http://svn.opendnssec.org/trunk opendnssec
```

The most recent version (at the time of writing version 1.0a1) can be found using:

```
svn co http://svn.opendnssec.org/tags/OpenDNSSEC-1.0a1 opendnssec
```

Building & Installing

Enter the opendnssec-directory:

```
cd opendnssec
```

```
sh autogen.sh
```

```
./configure \
    --prefix=/usr/local/opendnssec \
    --sysconfdir=/etc \
    --localstatedir=/var
```

You may also need some other options to configure, use the following command to find out which:

```
./configure --help
```

One configured, build OpenDNSSEC using:

```
make
```

… and install using …

```
make install
```

If the build fails it might be because of a missing software dependency. Please read the error messages carefully.

A detailed installation guide including the dependencies can be found here: http://trac.opendnssec.org/wiki/Signer/Install

Post-installation

Depending on operating system, there may be a few additional steps required after installation.

Linux Users

Linux users need to rebuild the dynamic linker caches. To do this, issue the command:

```
sudo ldconfig [library-path [library-path ...]]
```

If OpenDNSSEC or any of the pre-requisites were installed in non-standard directories, the list of library paths should be specified as arguments on the command line.

Configuration

The default configuration installed during the installation contains good default values for anybody who just want to sign their domains with DNSSEC. There are four configuration files for the basic OpenDNSSEC installation. First off there is the SoftHSM configuration which you need if you don't have a Hardware Security Module. For OpenDNSSEC itself you have *conf.xml* which is the overall configuration of the system, *kasp.xml* which contains the policy of signing and then there is *zonelist.xml* where you list all the zones that you are going to sign.

All date/time durations in the configuration files are specified as defined by ISO 8601.

SoftHSM / softhsm.conf

The file softhsm.conf lists all the slots in the virtualized HSM. The default location of the SoftHSM configuration file is /etc/softhsm.conf, but you can override the default location of the file using the environment variable `SOFTHSM_CONF`.

```
export SOFTHSM_CONF=/home/user/config.file
```

The file itself lists the slots like this:

```
0:/home/user/my.db
# Comments can be added
4:/home/user/token.database
```

The first number is the slot number, and after the colon is the path of a SQLite database where SoftHSM stores its keys. You only need one slot for basic OpenDNSSEC use.

To finish the setup of SoftHSM you also need to initialize the token:

```
softhsm --init-token --slot 0 --label "OpenDNSSEC"

Type in SO PIN and user PIN.
```

Use the label and PIN you used here in your conf.xml in conf.xml (also with the full path to the SoftHSM library) in order to use SoftHSM with OpenDNSSEC.

OpenDNSSEC Configuration - conf.xml

The overall configuration of OpenDNSSEC is defined by the contents of the file /etc/opendnssec/conf.xml. It defines such information as paths and database connections, and also includes a list of configured security modules.
In this configuration file you specify logging facilities (only syslog is supported now), system paths, key repositories and the database where all key and zone information is stored.

OpenDNSSEC stores its keys in *repositories*. There must be at least one repository, although the system can be configured to run with multiple repositories. There are a number of reasons for running with multiple repositories, including:
- Temporarily running with multiple repositories whilst a switch is made from one repository to another, e.g. when replacing hardware.
- The chosen type of repository has a limit on the number of keys that can be stored, and multiple repositories are needed to handle the chosen number of keys.
- Different types of repository are needed for different security levels, e.g. a key-signing key may require a higher level of security than a zone-signing key.

The Enforcer configuration in conf.xml specifies values that are not part of the policies described in kasp.xml.

The Signer configuration specifies the location of the working directory and how many threads to use when signing zones.

A full annotated example of this configuration file is to be found in the wiki:
http://trac.opendnssec.org/wiki/Signer/Configuration/ExampleConfXml

Key and Signature Policy - kasp.xml

kasp.xml - found by default in /etc/opendnssec - is the file that defines policies used to sign zones. (KASP stands for "Key and Signature Policy".) Each policy comprises a series of parameters that define the way the zone is signed. The hierarchical nature of XML allows the grouping of parameters into logical blocks.

You can have more than one policy, one for each zone if you like. A policy is referred to by a policy name in for example the zonelist.xml configuration file.

A policy have different parts separated into Signatures for signature related parameters, Denial for using NSEC or NSEC3, Keys for defining what keys you want to use and its lifetimes and Zone for zone file distribution parameters. For automatic key rollovers you also need to specify information about the parent zone in the Parent part.

More detailed information about configuring all the parameters in kasp.xml can be found on the wiki here:
http://trac.opendnssec.org/wiki/Signer/Configuration/ExampleKaspXml

The list of zones - zonelist.xml

The zonelist.xml file is used when first setting up the system, but also used by the signer_engine when signing zones. Within the ZoneList tag you list any number of zones you have. Each zone is using a policy as defined in kasp.xml. The SignerConfiguration is a file generated by the Enforcer to give to the signer with the current keys and parameters. The current available Adapters is the Input and Output files. The input files is the unsigned zone (currently without BIND-style directives) and the output file is the signed zone after signing.

```
<?xml version="1.0" encoding="UTF-8"?>

<ZoneList>
        <Zone name="example.se">
                <Policy>default</Policy>
                <SignerConfiguration>/var/opendnssec/signconf/example.se.xml</
SignerConfiguration>
                <Adapters>
                        <Input>
                                <File>/var/opendnssec/unsigned/example.se</File>
                        </Input>
                        <Output>
                                <File>/var/opendnssec/signed/example.se</File>
                        </Output>
                </Adapters>
        </Zone>
</ZoneList>
```

You must have a zonelist.conf for the system to start. You can add zones after starting the system by using the ksmutil command line utility, which will also make modifications to this file.

Running OpenDNSSEC

It should not matter in which order you start the different daemons in OpenDNSSEC, but our recommendation is that you first start *signer_engine*, and then *keygend* and last *communicated*. Before you run the system for the first time you must import your kasp.xml into the KASP Enforcer database. See the *ksmutil* command below.

In this alpha release all directories are prepared by the build script and are set to be owned by root, so all commands in the default configuration must also be run by root.

ksmutil

You need a way to interact to the KASP Enforcer, for example to add and remove zones that are handled by OpenDNSSEC. The *ksmutil* utility provides a number of commands to make this easy, and all commands are invoced on the unix command line.

You must use the `setup` before you ever run any sub-system in OpenDNSSEC. This reads the configuration kasp.xml och imports these settings into the KASP Enforcer database. The `setup` command **deletes the current content of the database!**

If you make any changes to kasp.xml these changes must be imported into the database. Use the `update` command to do this.

To add a zone to be handled by the KASP Enforcer, use the `addzone` command. This command need another parameter for which policy to use, and another two parameters for which paths to use for input and output. An example of use:

```
ksmutil addzone example.se default /var/example.se /var/
example.se.signed
```

The command `delzone` is simpler and needs no further parameters but the name of the zone.

Complete list of commands:

```
mask$~>ksmutil -h
usage: ksmutil [-f config_dir] setup [path_to_kasp.xml]
        Import config_dir into a database (deletes current
contents)
usage: ksmutil [-f config_dir] update [path_to_kasp.xml]
        Update database from config_dir
usage: ksmutil [-f config_dir] addzone zone [policy]
[path_to_signerconf.xml] [input] [output]
        Add a zone to the config_dir and database
usage: ksmutil [-f config_dir] delzone zone
        Delete a zone from the config_dir and database
usage: ksmutil [-f config_dir] listzone
        List zones from the zonelist.xml in config_dir
```

```
usage: ksmutil export [policy]
        export all policies [or named policy] to xml
usage: ksmutil [-f config_dir] rollzone zone [KSK|ZSK]
        Rollover a zone (may roll all zones on that policy)
usage: ksmutil [-f config_dir] rollpolicy policy [KSK|ZSK]
        Rollover all zones on a policy
```

signer_engine

This is the component that makes all the signing. At start it reads zonelist.xml and goes through all zones to sign them if needed.

```
mask$~>/usr/local/sbin/signer_engine
Python engine proof of concept, v 0.0002 alpha
Zone list updated: 0 removed, 3 added, 0 updated
output redirected to syslog
```

This daemon does not need any command line arguments to run.

keygend

The key generator daemon creates keys if needed by the KASP Enforcer sub-system.

communicated

The `communicated` program maintains keys according to policy (using libksm) for the Signer. It also creates signer configuration XML-files that the signer uses when signing the zones.

signer_engine_cli

The *signer_engine_cli* provides a Command Line Interface to the *signer_engine*. There is a number of commands you give to signer_engine. If you start the CLI without any command line parameters you enter a shell where you can issue commands:

```
mask$~>signer_engine_cli
connecting to /var/run/opendnssec/engine.sock
cmd> help
 Commands:
zones           show the currently known zones
```

```
sign <zone>      schedule zone for immediate (re-)signing
clear <zone>     delete the internal storage of the given
                 zone name. All signatures will be
                 regenerated on the next re-sign.
queue            show the current task queue
flush            execute all scheduled tasks immediately
update <zone>    check for changed zone conf xml file, if
                 <zone> is not given all zones are checked
stop             stop the engine
verbosity <nr>   set verbosity (notimpl)
```

The same commands can be passed as command line arguments in your unix shell.

In short, the *sign* commands tell the signer to immediately queue up a zone for re-signing. This is useful if you have made changes to your zonefile and you immediately want it signed for further distribution to your nameservers. *Clear* tells the signer to renew all signatures on the given zone. The *queue* command lists all the upcoming tasks currently in the queue.

hsmutil

The *hsmutil* utility is designed to interact with your HSM. With it you can list all keys in a given repository, and you can also use it to create or delete keys.

You can also use *hsmutil* to extract the public part of a key for publishing. In order to know what key to publish you need to look in your signer configuration file for your zone, for exampel /var/opendnssec/signconf/exempel.se.xml. This file lists all keys used for signing and inclusion, so you need to find the right key you want to publish. The you use the content of the Locator-tag and give it do hsmutil like this:

```
mask$~>hsmutil dnskey c562d0e07aaffb4930b2738670acc369 exempel.se

exempel.se.     3600    IN      DNSKEY  256 3 5
AwEAAb6ApPIehd3FiNt/2YihqzQUxZhS2Be/WVdy5b3meCjU8vaL6ec/oUN/
q2z2GV698BbsXH9Hx0ye316BiXn7cNIIyAgeQntqzohskwYMoCKwGIMjKATpFmk8a
ALWKmwUEfHW03QPHK0vY7hppBP0qw4BjDKeybvcRLXoLKZh/
MjZZoAHbybRctklbIL+5TWViOsJGrM2rB9YWp1uAoH+duEe/jDU0oOcE9E5QByUa/
13voFNDTqnSY3GS39Xpx22cpY3Rp4qs7hXujOIuyHVC4i43vAI/
PLyQuOmWDRDI1VgZcsOgSdx3lfNYGP8pwinwNVBHmuzoPIl/TkqzzTo1LU= ;{id
= 25861 (zsk), size = 2048b}
```

What you get in return is the DNSKEY in BIND-format, which can be converted to a DS record and included by a parent zone.

kasp_auditor
The Auditor part of the systems runs an audit of the zones in the system to see if the signer complies to what the policy mandates. It can be issued automatically after each resigning of a zone and will stop the signed zone file from being distributed. This component is still under development. To try it you can add the Audit-tag to your zone policy (kasp.xml).

Other HSMs

The main purpose of a HSM is to safeguard cryptographic key material and/or to speed up cryptographic operations (a HSM can have either one or both of these purposes). The price of an HSM range from about €50 to €30000 depending on security level, speed and storage capabilities. OpenDNSSEC uses the PKCS#11 interface for all cryptographic operations with security modules. A potential HSM for use with OpenDNSSEC must provide PKCS#11 and the cryptographic algorithms for use with DNSSEC.

The OpenDNSSEC project has developed a couple of tools to evaluate HSMs. The tool `hsmspeed` does performance testing on your HSM. This is also useful to find out at what speed you can get from SoftHSM on your CPU. This tool is not distributed as a part of the OpenDNSSEC distribution, but it can be found in the source tree under libhsm.

The `hsmbully` tool test your HSM for compliance with OpenDNSSEC. This tool is not distributed with OpenDNSSEC either, but can be found here in the SVN repository: http://trac.opendnssec.org/browser/trunk/hsmbully

The `hsmutil` tool is described above as it is part of the OpenDNSSEC project. It can be used to interact with your HSM.

OpenDNSSEC maintenance

Logs

Currently all logging is handled by syslog. Other logging methods may come later.

There is a lot of output from the daemons of which a lot of things right now are for debugging. We might reduce the amount of logging for later versions of OpenDNSSEC.

Update zones

When you update the content of an unsigned zone you can schedule it for immediate resigning using the signer_engine_cli command like this:

```
mask$~>signer_engine_cli sign exempel.se
connecting to /var/run/opendnssec/engine.sock
Zone scheduled for immediate resign
```

Publishing the keys

Please refer to the usage of *hsmutil* to see an example on how to publish a key. You only need to publish a key (to the parent or to interested parties) when you create a new KSK.

Key rollovers

To roll the keys for a zone you use the `ksmutil` command like this:

```
ksmutil rollzone example.se KSK
```

This will roll the KSK key in a timely manner following the policy used for the zone example.se. If you want to roll the Zone Signing Key use the parameter ZSK instead.

You can also roll all the keys for zones which have a certain policy. This can be useful if you want to move all keys from one key store to another.

```
ksmutil rollpolicy default KSK
```

Updating the policy

When you make changes to a policy or add a new policy in kasp.xml you use the `ksmutil update` command to update the changes in the database.
TODO: what happens when you change different things?

Adding and removing zones

`ksmutil` updates the zonelist.xml automatically when adding and removing zones. (need write permissions!)

Reporting bugs

OpenDNSSEC is currently only released as an alpha version. There might be bugs, unexpected behavior and undocumented features. If you encounter anything strange in either the documentation, the code or in the commands you are using, feel free to post a message to the opendnssec-user mailing list. You can subscribe to it here: http://lists.opendnssec.org/mailman/listinfo/opendnssec-user

If you have a bug report, please enter it into our *trac*-system: http://trac.opendnssec.org/newticket

Be sure to include all the error messages you get, the versions of the other libraries you are using and what actions you did to trigger the error message.