OpenDNSSEC Initial Deployment Guide DRAFT

W. Matthijs Mekking,*NLnet Labs*

August 29, 2014

Abstract

OpenDNSSEC is a policy-based zone signer that automates the process of keeping track of DNSSEC [1], [3], [2] keys and the signing of zones. The goal of the project is to make DNSSEC easy to deploy. The software has a lot of configuration options that can be tweaked but that requires some knowledge about DNSSEC. Some users might just want to deploy DNSSEC without knowing what is under the hood. If you are that person, this deployment guide will help you set up OpenDNSSEC and provide simple guidelines for adding and removing zones, as well as some other quick guides.

Contents

1 Starting OpenDNSSEC for the first time

OpenDNSSEC [4] will run on most Linux, BSD and Solaris operating systems. The community provides binary packages for several platforms so that might make installation even more easy. This guide assumes those packages are not available.

1.1 Dependencies

If you have found a nice system to run OpenDNSSEC on, it is time to install its dependencies. OpenDNSSEC relies on a database backend and currently supports MySQL and SQLite. **MySQL** is recommended because SQLite doesn't scale well and has some known locking issues. Furthermore, OpenDNSSEC depends on:

- **ldns**, version 1.6.12 and up with the exceptions of 1.6.14 and 1.6.15;
- **libxml2**, libxml2-dev, libxml2-utils, version 2.6.16 and up.

1.1.1 SoftHSM

OpenDNSSEC also requires a Hardware Security Module (HSM). These things may be expensive, but you can use **SoftHSM**, a software-only implementation of an HSM, as an alternative.

SoftHSM depends on **Botan** (a cryptographic library) version 1.8.5 or greater, and **SQLite** version 3.3.9 or greater. Install SoftHSM with:

```
$ tar -xzf softhsm-X.Y.Z.tar.gz
$ cd softhsm-X.Y.Z
$ ./configure
$ make
$ make install
```

By default, the binary will be installed in /usr/local/bin/ and the configuration is expected to be at /etc/softhsm.conf. Open the file and specify a slot for OpenDNSSEC. For example:

```
# SoftHSM slots
0:/var/lib/softhsm/slot0.db
```

The token database does not exist at this stage. You need to initalize it with:

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

When prompted, fill in a SO (Security Officer) PIN and user PIN. Remember it, you will need to configure it for OpenDNSSEC. The SO PIN can be used to re-initalize the token. The user PIN is handed out to OpenDNSSEC. If your company does not have a SO, just pick the same PIN for both roles.

Make sure OpenDNSSEC has permission to access the token database.

```
$ chown opendnssec /var/lib/softhsm/slot0.db
$ chgrp opendnssec /var/lib/softhsm/slot0.db
```

1.2 Installation

Run these commands to install OpenDNSSEC:

```
$ tar -xzf opendnssec-X.Y.Z.tar.gz
$ cd OpenDNSSEC-X.Y.Z
$ ./configure
$ make
$ make install
```

By default, the binaries will be installed in /usr/local/bin/ and /usr/local/sbin/. The configuration files are located in the /etc/opendnssec/ directory. The working directories are under /var/opendnssec/.

1.3 Configuration

The default configuration installs good default values for anyone who just wants to sign their domains with DNSSEC. There are four configuration files for the basic OpenDNSSEC installation:

- **conf.xml** which is the overall configuration of the system;
- **kasp.xml** which contains the policy of signing;
- **zonelist.xml** where you list all the zones that you are going to sign;
- **addns.xml** (per zone, optional) for zone transfers.

For now, we only need to edit **conf.xml** because we need to configure the HSM, in our case SoftHSM. Make the Repository part look like:

```
<Repository name="SoftHSM">
    <Module>/usr/local/lib/libsofthsm.so</Module>
    <TokenLabel>OpenDNSSEC</TokenLabel>
    <PIN>XXXX</PIN>
    <SkipPublicKey/>
</Repository>
```

Here, XXXX is the user PIN you filled in 1.1.1.

1.3.1 Key and Signing Policy (KASP)

The default KASP is provides good standard values for signing any zone. However, if you want to change any value, you can add a new policy here, or change values in an existing policy. For example, if your zones use the YYYYMMDDXX format for SOA SERIAL values, change the **Serial** parameter in **kasp.xml** from unixtime to **datecounter**:

```
<Zone>
    <PropagationDelay>PT9999S</PropagationDelay>
    <SOA>
        <TTL>PT3600S</TTL>
        <Minimum>PT3600S</Minimum>
        <Serial>datecounter</Serial>
    </SOA>
</Zone>
```

For full descriptions about all the KASP parameters, see the OpenDNSSEC Wiki [5].

You are now ready to start OpenDNSSEC. The very first time you need to setup the database. There is a control script that starts up two daemons: **ods-enforcerd** that takes care of the key management, and **ods-signerd** that is the actual signer. Run:

```
$ ods-ksmutil setup
*WARNING* This will erase all data in the database; are you sure? [y/N] y
$ ods-control start
```

Congratulations! You are now running OpenDNSSEC. Logs are going to syslog. The setup has imported the two default Key And Signing Policies (KASP), default and lab. However, no zones are imported yet.

2 Adding a zone

Because we did not edit the zone list **zonelist.xml**, we started OpenDNSSEC with no zones to sign. It is time to add the zones you want to sign. This can be done with:

```
$ ods-ksmutil zone add -z example.com
```

This will add the zone **example.com** to OpenDNSSEC with the default KASP. Also by default, the signing will be file based. The signer expects the unsigned file to be at /var/opendnssec/unsigned/example.com and puts the signed file at /var/opendnssec/signed/example.com. You can use different paths with **-i** (input) and **-o** (output). You can use a different policy with **-p** (policy).

If you want to use DNS zone transfers for input and output, you can set the type of adapter to DNS, **-j** for input and **-q** for output. You will need to set the input and output files to the zone transfer configuration file **addns.xml**, like this:

```
$ ods-ksmutil zone add -z example.com -j DNS -q DNS \
    -i /etc/opendnssec/addns.xml -o /etc/opendnssec/addns.xml
```

How to edit **addns.xml** for zone transfers is described in Section 2.1.

After adding (and removing) zones, you might want to export the new zonelist to reflect the changes:

```
$ ods-ksmutil zonelist export > zonelist.xml
```

The enforcer will only check once an hour to see if there is work to do, so if you want your zone to be signed right now, you need to notify the daemon:

```
$ ods-ksmutil notify
```

And soon you will see the signed zone be written in the /var/opendnssec/signed/ directory. Notify your name server of the new zonefile so the zone will also become visible in the DNS. You can configure a notify command in **conf.xml** to automatically notify your name server of new zones. For example:

```
<Configuration>
    ...
    <Signer>
    ...
        <NotifyCommand>nameserver_control_program reload %zone</NotifyCommand>
    </Signer>
</Configuration>
```

Here, %zone will be replaced with the name of the zone that has been updated, and %zonefile (not used in example) will be replaced with the name of the signed zonefile.

2.1 OpenDNSSEC as a bump-in-the-wire

If in Section 2 you added a zone with DNS adapters instead of working on files, instead of pointing the input and output to the filenames of the unsigned and signed zones, you need to put in the zone transfer configuration file **addns.xml**. Here, you can set up your primary name server addresses, ports and TSIG keys (Inbound), and the same for the secondary name servers (Outbound). Replace the example values in **addns.xml.sample** installed in /etc/opendnssec/ with your own servers and keys and rename it to **addns.xml**. Also **conf.xml** needs a change: we now need a socket that listens to DNS traffic:

```
<Configuration>
    ...
    <Signer>
    ...
        <Listener>
            <Interface><Address>127.0.0.1</Address><Port>53</Port></Interface>
            <Interface><Address>::1</Address><Port>53</Port></Interface>
        <Listener>
    </Signer>
</Configuration>
```

The above values are also the defaults.

OpenDNSSEC can now sign incoming zone transfers (full and incremental) and also reply to SOA, AXFR and IXFR requests.

3 Publish the DS

If you list the keys, you might see that the key signing key (KSK) is not yet active:

```
$ ods-ksmutil key list
Zone:           Keytype:      State:    Date of next transition:
example.com.    KSK           publish   2014-04-25 03:51:26
example.com.    ZSK           active    2014-07-23 13:51:26
```

This is because we still need to submit the DS to the parent. The DS is a record that is derived from the KSK and is published in the parent zone. This is used to build a secure chain of trust from the root zone to your zone.

In the example, OpenDNSSEC expects this to happen at 3:51 AM on the 25th of April 2014. This is 14 hours after initial signing! Why is that? The default policy has a very conservative propagation delay for the name servers: 12 hours. Take an additional hour for the TTL and one more for the publish safety parameter and it adds up. Long propagation delay or not, you would have to wait nevertheless because in order to make sure your zone does not go bogus, we will have to respect a publish safety duration and the TTL (in this case derived from the SOA MINIMUM).

If OpenDNSSEC is ready, the date of next transition will tell you **waiting for ds-seen**. We can now submit the DS to the parent. How you do that depends on your registrar. Usually this can be done via e-mail or through a web interface. Retrieve the DNSKEY or DS with:

```
$ ods-ksmutil key export

;ready KSK DNSKEY record:
example.com. 3600 IN DNSKEY 257 3 8 Aw...

$ ods-ksmutil key export -d

;ready KSK DS record (SHA1):
example.com.. 3600 IN DS 42112 8 1 8aea...

;ready KSK DS record (SHA256):
example.com. 3600 IN DS 42112 8 2 a674...
```

If the DS shows up in the parent zone at all parent name servers, it is safe to run the **key ds-seen** command. This command requires the keytag of the key in question. You can see from the DNSKEY and DS records this is 42112 in this example:

```
$ ods-ksmutil key ds-seen -z example.com -x 42112
```

Now if you run **key list** again, you will see the KSK is now also active. The chain-of-trust is set up!

4 Migrate a zone to OpenDNSSEC

It is possible that you already have your zones DNSSEC signed but want to migrate to OpenDNSSEC. How to migrate to OpenDNSSEC really depends on how your current solution looks like. This quick guide assumes that the zone is already under your operative domain.

4.1 Export the keys

First of all, you need to export the keys and get them into your HSM, in this case SoftHSM. The BIND .private-key file can be convert into the PKCS#8 file format using the tool available with SoftHSM. If you have another file format, then OpenSSL probably can help you to convert it into the PKCS#8 file format.

```
$ softhsm-keyconv --topkcs8 --in Kexample.com.+008+42112.private --out ksk.pem
$ softhsm-keyconv --topkcs8 --in Kexample.com.+008+7938.private --out zsk.pem
```

If keys are already in a SoftHSM instance, you can export them like this:

```
$ softhsm --slot 0 --pin XXXX --id b528... --export ksk.pem
$ softhsm --slot 0 --pin XXXX --id 893b... --export zsk.pem
```

The IDs above are shortened for readability. Copy the files to the new system. The PKCS#8 files can now be imported into the SoftHSM token on that sytem:

```
$ softhsm --import ksk.pem --slot 0 --pin XXXX --label KSK --id b528...
$ softhsm --import zsk.pem --slot 0 --pin XXXX --label ZSK --id 893b...
```

4.2 Add the zone to OpenDNSSEC

It is important that you stop OpenDNSSEC before adding the zone. Otherwise OpenDNSSEC will create new keys for the zone on its own, not using our imported keys. Add the zone with **ods-ksmutil**.

```
$ ods-control stop
$ ods-ksmutil zone add -z example.com
```

There are two elements that are important to keep in sync:

- **Key algorithms and lengths** If the key algorithms in the policy does not match the algorithms and lengths of the keys on the previous system, you will break the DNSSEC chain-of-trust. Make sure that the policy used and described in **kasp.xml** has the same values for key algorithms and lengths (//Keys/KSK/Algorithm and //Keys/ZSK/Algorithm).
- **SOA serial** OpenDNSSEC maintains the SOA serial. Change the unsigned zone in such a way that the SOA serial is higher than in the current zone. This is especially important if your KASP uses **counter** as serial format (//Zone/SOA/Serial).

Import the keys into OpenDNSSEC. This will require quite some parameters:

1. **-k or –cka-id** The CKA_ID of the key, get it with **ods-ksmutil key list -v** on the previous system.
2. **-r or –repository** The repository where the key is stored, from **conf.xml**.
3. **-z or –zone** The zone which should use the key.
4. **-b or –bits** The key length, get it from the **kasp.xml** on the old system.
5. **-g or –algorithm** The key algorithm, also from the **kasp.xml** on the old system.
6. **-e or –keystate** The starting key state, get it with **ods-ksmutil key list** on the previous system. If you did not run OpenDNSSEC on the previous system, make sure you are not currently rolling keys and use the keystate **active**.

7. **-t or –keytype** The type of the key, KSK or ZSK.
8. **-w or –time** The date or datetime the key entered the key state or an arbitrary time.

In our example, keys are imported as follows:

```
$ ods-ksmutil key import -k b528... -r SoftHSM -z example.com \
                         -b 2048 -g 8 -e ACTIVE -t KSK -w 20140425
$ ods-ksmutil key import -k 893b... -r SoftHSM -z example.com \
                         -b 1048 -g 8 -e ACTIVE -t ZSK -w 20140424
```

Now copy the unsigned zone data, or make sure the DNS adapters are properly configured and start OpenDNSSEC again:

```
$ ods-control start
```

The zone is now successfully migrated to OpenDNSSEC!

5 Feedback

I hope this guide turns out to be useful. If you have feedback or still run into trouble, you can subscribe and mail to `opendnssec-user@lists.opendnssec.org`.

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *DNS Security Introduction and Requirements*. RFC 4033 (Proposed Standard), March 2005. `http://www.ietf.org/rfc/rfc4033.txt`.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *Protocol Modifications for the DNS Security Extensions*. RFC 4035 (Proposed Standard), March 2005. `http://www.ietf.org/rfc/rfc4035.txt`, (Updated by RFC 4470).

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *Resource Records for the DNS Security Extensions*. RFC 4034 (Proposed Standard), March 2005. `http://www.ietf.org/rfc/rfc4034.txt`, (Updated by RFC 4470).

[4] OpenDNSSEC web page. `https://www.opendnssec.org/`.

[5] OpenDNSSEC Documentation. `https://wiki.opendnssec.org/`.