Zone signing with OpenDNSSEC - part 1

1. Initialize the Software "Hardware Security Module"

    Start by becoming root for this session (or use sudo when required)

    ```
    $ sudo -s
    #
    ```

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

    (use '1234' for both questions below):

    ```
    The SO PIN must have a length between 4 and 255 characters.
    Enter SO PIN: ****
    The user PIN must have a length between 4 and 255 characters.
    Enter user PIN: ****
    The token has been initialized.
    ```

    ```
    # softhsm --show-slots
    ```

    Create configuration files for OpenDNSSEC by making a copy
    of the samples distributed with the package:

    ```
      # cd /usr/local/etc/opendnssec
      # cp kasp.xml.sample kasp.xml
      # cp conf.xml.sample conf.xml
      # cp addns.xml.sample addns.xml
      # cp zonelist.xml.sample zonelist.xml
      # chmod 644 *xml
    ```

2. Change the default Policy to use NSEC instead of NSEC3:

    Edit /usr/local/etc/opendnssec/kasp.xml

    Find this section, and remove all the lines from <NSEC3> ... </NSEC3>

    ```
    <NSEC3>
        <!-- <OptOut/> -->
        <Resalt>P100D</Resalt>
        <Hash>
            <Algorithm>1</Algorithm>
            <Iterations>5</Iterations>
            <Salt length="8"/>
        </Hash>
    </NSEC3>
    ```

    ... and replace them with this single line:

    ```
    <NSEC/>
    ```

    Save & exit.

    Also, set the correct path for the libsofthsm.so in the conf.xml:

    Change

```
<Module>/usr/local/lib/libsofthsm.so</Module>
```

to

```
<Module>/usr/local/lib/softhsm/libsofthsm.so</Module>
```

    In the same file, find the line:

```
<Datastore><SQLite>/usr/local/var/opendnssec/kasp.db</SQLite></Datastore>
```

    Remove it, and instead, add:

```
<Datastore>
    <MySQL>
        <Host port="3306">localhost</Host>
        <Database>opendnssec</Database>
        <Username>root</Username>
        <Password></Password>
    </MySQL>
</Datastore>
```

3. Start MySQL and create the database

    Edit /etc/rc.conf, and add:

    mysql_enable="YES"

    Save and exit the file, then run:

    # service mysql-server start

    You should see:

    Starting mysql.

    Create the database:

    # echo "create database opendnssec" | mysql

4. Initialize the KSM

    # ods-ksmutil setup

```
*WARNING* This will erase all data in the database; are you sure? [y/N] y
Enter password:
```

    Just press <ENTER> when asked for the password. You will then see:

```
    zonelist filename set to /usr/local/etc/opendnssec/zonelist.xml.
    kasp filename set to /usr/local/etc/opendnssec/kasp.xml.
    Repository SoftHSM found
    No Maximum Capacity set.
    RequireBackup NOT set; please make sure that you know the potential problems of
using keys which are not recoverable
    INFO: The XML in /usr/local/etc/opendnssec/conf.xml is valid
    INFO: The XML in /usr/local/etc/opendnssec/zonelist.xml is valid
    INFO: The XML in /usr/local/etc/opendnssec/kasp.xml is valid
    Policy default found
```

5. Install a copy of the unsigned zone for OpenDNSSEC to sign

Earlier, we made a backup copy of our zone, before it was signed by BIND9. We are going to use that backup copy now and make it available to OpenDNSSEC.

```
# cd /etc/namedb/master
# cp mytld.backup /usr/local/var/opendnssec/unsigned/mytld
```

Increment the serial in the mytld file, so that it's up-to-date (YYYYMMDDXY).

6. Add the zone to OpenDNSSEC's database:

```
# ods-ksmutil zone add --zone mytld

zonelist filename set to /usr/local/etc/opendnssec/zonelist.xml.
Imported zone: mytld
```

7. Start OpenDNSSEC!

Add this to /etc/rc.conf

```
opendnssec_enable="YES"
```

Save the file and exit.

Now, start the service:

```
# service opendnssec start
```

You will see:

```
Starting enforcer...
OpenDNSSEC ods-enforcerd started (version 1.4.3), pid 2923
Starting signer engine...
OpenDNSSEC signer engine version 1.4.3
Engine running.
```

```
# ps ax | grep ods

41588  ??  SsJ    0:00.11 /usr/local/sbin/ods-enforcerd
41593  ??  SsJ    0:00.07 /usr/local/sbin/ods-signerd
```

8. Check that the zone is signed

```
# ls -l /usr/local/var/opendnssec/signed

-rw-r--r--  1 root  wheel  2621 Feb 19 09:10 mytld
```

Take a look at the contents of the zone - note the key ids for the KSK and ZSK.

If for some reason, you don't see a file in this directory (/usr/local/var/opendnssec/signed/), then force the signer to sign:

```
# ods-signer sign mytld
```

9. Moment of reflection

Ok, so now the zone is signed with OpenDNSSEC - do notice that the zone was signed, but you didn't issue any commands to generate keys.

List the keys currently managed by OpenDNSSEC:

```
# ods-ksmutil key list
```

```
Keys:
Zone:                           Keytype:      State:    Date of next transition:
mytld                           KSK           publish   2014-03-21 04:25:30
mytld                           ZSK           active    2014-03-21 04:32:30
```

Notice that two keys have just been created by OpenDNSSEC, on the fly.

But BIND is still loading the zone that was signed earlier (either manually or using the inline signer) - can we just modify the named.conf definition and point to the signed zone instead ?

Which KSK is currently being used ? And which DS record is published in the parent zone ?

Would the resolvers be able to verify the signatures on the zone signed with OpenDNSSEC ? Why not ? What would you have to do for it to work (there are several possible answers)

If you don't care about the validation problem, then you can proceed with the rest of this lab.

10. Tell BIND to load the new zone

Modify /etc/namedb/named.conf, and change the zone definition for "mytld" so it looks like this (REMOVE auto-dnssec, etc...)

```
zone "mytld" {
    file "/usr/local/var/opendnssec/signed/mytld";  // <--- Change path
    type master;
    key-directory "/etc/namedb/keys";   // <--- Remove if there
    auto-dnssec maintain;               // <--- Remove if there
    inline-signing yes;                 // <--- Remove if there
};
```

Now, BIND is back to being a "passive" nameserver that doesn't sign the zone - it just serves the zone signed by OpenDNSSEC.

Restart named:

```
# service named restart
```

Check the logs in /etc/namedb/log/general to make sure that the zone is loading correctly.

Now, validation will probably fail for those trying to look up data in your zone. Wait a few minutes, and try to lookup a record in your zone:

```
# dig @127.0.0.1 www.mytld +dnssec
```

What do you notice ?

11. OpenDNSSEC reload BIND

    Even better, you can have OpenDNSSEC tell BIND to reload the zone when
    it has been signed - like this, no need to manually reload.

    To do this, modify /usr/local/etc/opendnssec/conf.xml

    Find the lines:

```
<!--
                <NotifyCommand>/usr/sbin/rndc reload %zone</NotifyCommand>
-->
```

    ... remove the comments (the lines '<!--' and '-->') before and after.

    Save the file, and restart OpenDNSSEC:

```
    # ods-control stop
    ...
    # ods-control start
```

12. Export the DS, ready to upload:

    Verify the state of the KSK at this stage:

```
    # ods-ksmutil key list
```

    Note the state that the KSK is in.

    If it is still in publish state (see
    https://wiki.opendnssec.org/display/DOCS/Key+States#KeyStates-Publish for
    reference), then the key is, from OpenDNSSEC's point of view, not ready to
    be used, as it hasn't had time to propagate.

    You can still export the DS record, derived from the KSK:

```
    # ods-ksmutil key export --zone mytld --ds --keystate publish
```

    Note the warning!

```
WARNING: No active or ready keys seen for this zone. Do not load any DS records to
the parent unless you understand the possible consequences.
```

    Ok, let's send the DS to a file, so we can upload it:

```
    # ods-ksmutil key export --zone mytld --ds --keystate publish >/tmp/dsset-
mytld.
```

13. Upload the DS to the server

    If you're not using the web interface:

```
    # scp /tmp/dsset-mytld. sysadm@a.root-servers.net:
```

14. Notify the administrator!

    Ask the root operator to add the new DS to the root zone, and see
    how long it takes before validation starts working again for your zone.

... or if using the web interface:

Log into https://rzm.dnssek.org and fix the DS records by verifying
the "eyed" DS records and checking them and then clicking "Update".
After a few minutes for caches, the resolver should validate.
If you have problems, ask instructor to flush the cache.

15. What's with the keystate ?

Why is the key in Publish state ? Why is OpenDNSSEC reluctant to let us
use the key right away ?

Was it a good idea to upload the DS already ?

If you wait long enough, you will see this:

```
Keys:
Zone:                           Keytype:      State:    Date of next transition:
mytld                           KSK           ready     waiting for ds-seen
mytld                           ZSK           retire    2014-03-21 07:50:38
mytld                           ZSK           active    2014-03-21 07:54:38
```

In reality, we should have waited until the key was marked
as "ready" before we published the DS!

Why ? There was a risk the zone information wasn't fully propagated
(think secondaries and caches). Only once the key is marked as ready
is it safe to upload the DS. OpenDNSSEC uses the parameters in the
policy settings (kasp.xml) to determine this.

16. Informing OpenDNSSEC that the DS is seen in the parent zone

Once you have seen the DS in the parent zone, and the KSK is in the
"ready" state, then you can tell OpenDNSSEC about it.

```
# ods-ksmutil key list -v

Keys:
Zone:                           Keytype:      State:    Date of next transition
(to):  Size:   Algorithm:  CKA_ID:                          Repository:
Keytag:
mytld                           KSK           ready     waiting for ds-seen
(active)   2048    8           0c4f577032e04e2eb34163382a4524d7  SoftHSM
44096
mytld                           ZSK           active    2014-03-21 07:54:38
(retire)   1024    8           bbd9b3e14c3cbb0517d49f79985916bd  SoftHSM
57634
mytld                           ZSK           publish   2014-03-21 09:02:55 (ready)
1024    8           7982538186c1b77afe84e6875f3c7bda  SoftHSM
51991
```

-v gives you the key ids, which you will need for the next step.

Note the keyid of the KSK, which is in `ready` state.

Now, do:

```
# ods-ksmutil key ds-seen --zone mytld --keytag 44096
```

... where 44096 is the keyid of the KSK in the example above.

You will see:

```
Found key with CKA_ID 0c4f577032e04e2eb34163382a4524d7
Key 0c4f577032e04e2eb34163382a4524d7 made active
Notifying enforcer of new database...
Performed a HUP ods-enforcerd
```

Ok, look at the keys again:

```
# ods-ksmutil key list
```

Note that the KSK is now marked active.