DNSSEC

Integration Guide

gemalto
security to be free

Document Number: 007-011195-001, Rev C.

Release Date: August 2017

Contents

Preface

This document is intended to guide security administrators to install, configure, and integrate DNSSEC and BIND with SafeNet Luna Hardware Security Module (HSM).

Scope

This guide provides instructions for setting up a small test lab with BIND and DNSSEC running with Luna HSM for securing the SSL certificate private keys. It explains how to install and configure the software that is required for setting up BIND and DNSSEC Server while storing certificate private key on Luna HSM.

Document Conventions

This section provides information on the conventions used in this template.

Notes

Notes are used to alert you to important or helpful information. These elements use the following format:

NOTE: Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. These elements use the following format:

CAUTION: Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. These elements use the following format:

WARNING: Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.

Command Syntax and Typeface Conventions

Convention	Description
bold	The bold attribute is used to indicate the following: • Command-line commands and options (Type **dir /p**.) • Button names (Click **Save As**.) • Check box and radio button names (Select the **Print Duplex** check box.) • Window titles (On the **Protect Document** window, click **Yes**.) • Field names (**User Name:** Enter the name of the user.) • Menu names (On the **File** menu, click **Save**.) (Click **Menu** > **Go To** > **Folders**.) • User input (In the **Date** box, type **April 1**.)
italic	The italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
`Consolas`	Denotes syntax, prompts, and code examples.

Support Contacts

Contact Method	Contact Information	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017, USA	
Phone	US	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://supportportal.gemalto.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.	

1
Introduction

Overview

This document is intended to guide security administrators to install, configure and integrate ISC (Internet System Consortium) BIND (Berkeley Internet Name Domain) and OpenDNSSEC with SafeNet Luna Hardware Security Module.

BIND is by far the most popular and widely used DNS software on the Internet. It provides a robust and stable platform on top of which organizations can build distributed computing systems with the knowledge that those systems are fully compliant with published DNS standards. BIND supports the full DNSSEC standard.

OpenDNSSEC is a designated DNSSEC signer tool using PKCS#11 to interface with Hardware Security Modules. It automates the process of keeping track of DNSSEC keys and signing of zones. The Key storage and hardware acceleration is achieved using PKCS#11 device.

Understanding the DNSSEC

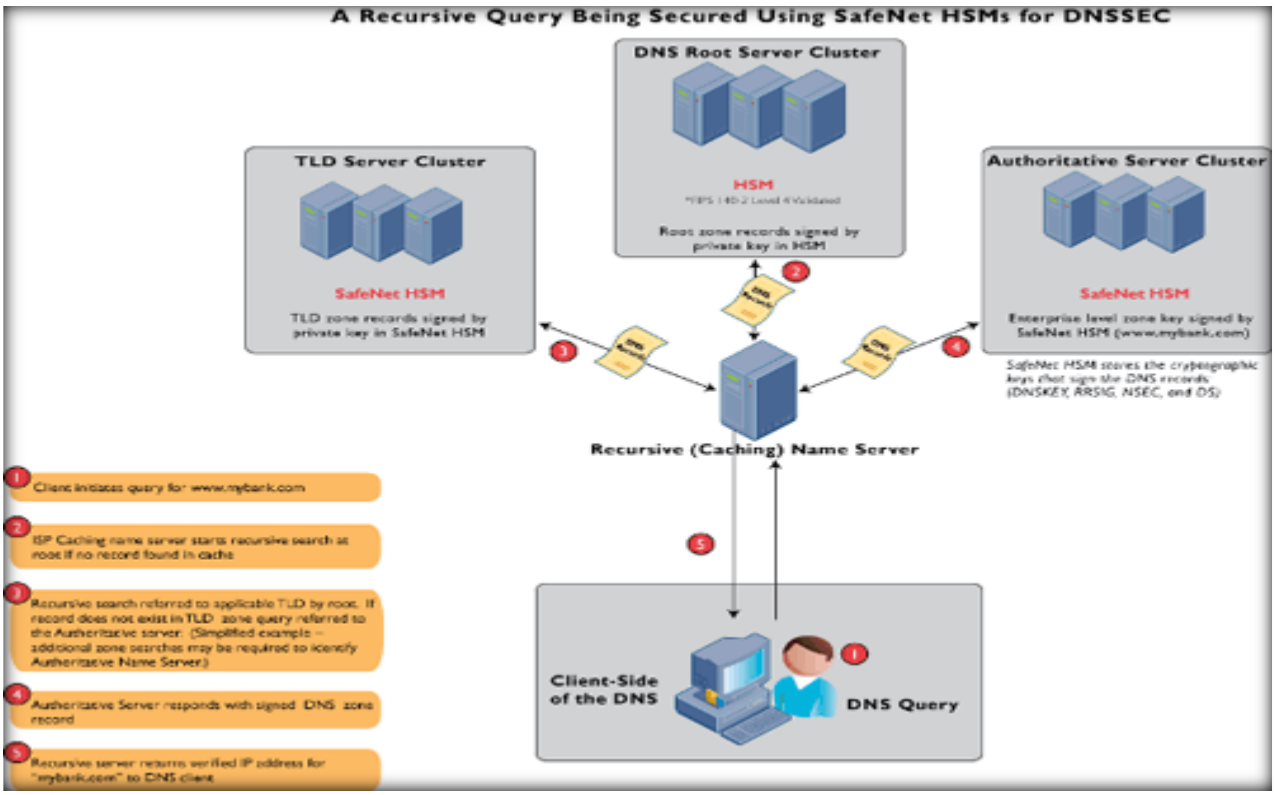
What is DNSSEC

DNSSEC is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol networks. It is a set of extensions to DNS which provide to DNS clients (resolvers) origin authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.

What role does a HSM play in DNSSEC

It is imperative that private DNSSEC signing keys are kept secure. By definition, the public key can be made widely available; it does not need to be secured. However, if the private key is compromised, a rogue DNS server can masquerade as the real authoritative server for a signed zone. This is where HSMs come into play. HSMs secure the DNS server so the generation of keys, the storing of the private key, and the signing of zones is performed on a DNS server that is physically secure and whose access is restricted to essential personnel only.

In addition SafeNet HSMs support key rollover functions, since DNSSEC keys do not have a permanent lifetime. The chances a key will be compromised, whether through accident, espionage, or cryptanalysis, increase the longer the key is used.



3rd Party Application Details

OpenSSL is a popular Open Source implementation of the SSL/TLS protocols. The project is managed by a worldwide community of volunteers. OpenSSL is the only free, full-featured SSL implementation currently available for use with the C and C++ programming languages. It works across most major platforms, including Microsoft Windows and Unix operating systems.

Supported Platforms

The following platforms are tested with SafeNet Luna HSM:

SafeNet Luna HSM (v7.0)

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	OpenSSL, DNSSEC, BIND
Red Hat Enterprise Linux 7 64-bit	7.0.0	7.0.0	7.0.1	OpenSSL v1.0.2 BIND v9.11.1 OpenDNSSEC v2.1.0

SafeNet Luna HSM (v5.x/6.x)

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	OpenSSL, DNSSEC, BIND
Red Hat Enterprise Linux 7 64-bit	6.3.0	6.3.0	6.10.9 6.27.0	OpenSSL v1.0.2 BIND v9.11.1 OpenDNSSEC v2.1.0
Debian Linux 6.0 (64 bit)	5.1	5.1	6.2.1	OpenSSL v0.9.8 BIND v9.9.2 OpenDNSSEC v1.3.12
RHEL 5.4 (32-bit / 64-bit)	5.0	5.0	6.0.7	OpenSSL v0.9.8 BIND v9.7 OpenDNSSEC v1.0.0
Solaris 10 SPARC (32-bit)	5.0	5.0	6.0.7	OpenSSL v0.9.8 BIND v9.7 OpenDNSSEC v1.0.0

Prerequisites

Configuring SafeNet Luna Network HSM 7.0

SafeNet Luna Network HSM allows to create Per-Partition Security Officer (PPSO) partition. HSM Administrator is not Security Officer (SO) for PPSO partitions. The HSM SO/Administrator elects to create a partition as PPSO-type, which creates an empty structure that is handed to the new owner, who initializes the partition to create the Partition Security Officer (PSO) role or identity for management functions. The PSO in turn creates the partition Crypto Officer (CO) to control client cryptographic operations on the partition.

Before you get started ensure the following:

- SafeNet Luna Network HSM appliance and a secure admin password.
- SafeNet Luna Network HSM, and a hostname, suitable for your network.
- SafeNet Luna Network HSM network parameters are set to work with your network.
- Initialize the HSM on the SafeNet Luna Network HSM appliance.

- Create and exchange certificates between the SafeNet Luna Network HSM and your Client system.
- Create a partition on the HSM that will be later used by DNSSEC.
- Register the Client with the partition. And run the "`vtl verify`" command on the client system to display a partition from SafeNet Luna HSM. The general form of command is "`C:\Program Files\SafeNet\LunaClient> vtl verify`" for Windows and "`/usr/safenet/lunaclient/bin/vtl verify`" for Unix.
- Initialize the Partition as mentioned in steps below for Password/PED based respectively

Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet Luna Network HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

Initialize the Partition SO and Crypto Officer Roles on a PW-Auth Partition

These instructions assume a password-authenticated SafeNet Luna Network HSM that has been initialized, and an application partition has been created, capable of having its own Security Officer.

- **Initialize the Partition SO role**

  Set the active slot to the created, uninitialized, application partition:
  Type **slot set -slot <slot number>**

  ```
  lunacm:> slot set -slot 0
  Current Slot Id:    0     (Luna User Slot 7.0.0 (Password) Signing With Cloning Mode)
  Command Result : No Error
  ```

  Initialize the application partition, to create the partition's Security Officer (SO).
  Type **partition init -label <partition label>**

  ```
  lunacm:> par init -label <part_label> –password <part_password>
           You are about to initialize the partition.
           All partition objects will be destroyed.
           Are you sure you wish to continue?
           Type 'proceed' to continue, or 'quit' to quit now -> proceed
  Command Result: No Error
  ```

- **Initialize the Crypto Officer role**

  a. The SO of the application partition can now assign the first operational role within the new partition.
  Type **role login -name Partition SO**.

  ```
  lunacm:> role login -name Partition SO
  ```

  b. Type **role init -name Crypto Officer**.

  ```
  lunacm:> role init -name Crypto Officer
  ```

  c. The application partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. Therefore, the SO must log out to allow the Crypto Officer to log in.
  Type **role logout**.

  ```
  lunacm:> role logout
  ```

Initialize the Partition SO and Crypto Officer Roles on a PED-Auth Partition

These instructions assume a PED-authenticated SafeNet Luna Network HSM that has been initialized, and an application partition has been created, capable of having its own Security Officer.

Take the following steps to initialize the PSO and CO roles:

- **Initialize the Partition SO role**

  Set the active slot to the created, uninitialized, application partition.
  Type **slot set -slot <slot number>**

```
lunacm:> slot set -slot 0
        Current Slot Id:    0     (Luna User Slot 7.0.0 (PED) Signing With Cloning Mode)
Command Result : No Error
```

Initialize the application partition, to create the partition's Security Officer (SO).
Type **partition init -label <partition label>**

```
lunacm:> par init -label <part_label>
        You are about to initialize the partition.
        All partition objects will be destroyed.
        Are you sure you wish to continue?
        Type 'proceed' to continue, or 'quit' to quit now -> proceed
        Please attend to the PED.
 Respond to SafeNet PED prompts...

Command Result : No Error
```

- **Initialize the Crypto Officer role**

  The SO of the application partition can now assign the first operational role within the new partition.

  Type **role login -name Partition SO**.

  Type **role init -name Crypto Officer**.

```
lunacm:> role init -name Crypto Officer
        Please attend to the PED.
Respond to SafeNet PED prompts...

Command Result: No Error
```

The application partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. Therefore, the SO must log out to allow the Crypto Officer to log in.
Type **role logout**.

Now, the Crypto Officer, or an application using the CO's challenge secret/password can perform cryptographic operations in the partition, as soon as the Crypto Officer logs in with **`role login -name Crypto Officer`**. However, the Crypto Officer can create, modify and delete crypto objects within the partition, in addition to merely using existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them.

NOTE: The black Crypto Officer PED key/Crypto Officer password is valid for the initial login only. You must change the initial credential on the key using the command **role changepw** during the initial login session, or a subsequent login. Failing to change the credential will result in a CKR_PIN_EXPIRED error while performing role-dependent actions.

Configuring SafeNet Luna Network HSM (v5.x/6.x)

Refer to the SafeNet Luna HSM documentation for installation steps and details regarding the configuration and setup of the box on UNIX systems. Before you get started ensure the following:

- SafeNet Luna Network HSM appliance and a secure admin password.
- SafeNet Luna Network HSM, and a hostname, suitable for your network.
- SafeNet Luna Network HSM network parameters are set to work with your network.
- Initialize the HSM on the SafeNet Luna Network HSM appliance.
- Create and exchange certificates between the SafeNet Luna Network HSM and your Client system.
- Create a partition on the HSM, remember the partition password that will be later used by DNSSEC.
- Register the Client with the partition. And run the `"vtl verify"` command on the client system to display a partition from SafeNet Luna Network HSM. The general form of command is `"C:\Program Files\SafeNet\LunaClient> vtl verify"` for Windows and `"/usr/safenet/lunaclient/bin/vtl verify"` for Unix.
- Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet Luna Network HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

NOTE: For Ped based SafeNet Luna HSM make sure ProtectedAuthenticationPathFlagStatus is set to '1' in Misc Section of Chrystoki.conf file.

SafeNet Luna HSM Configuration Settings

The Luna Client configuration file located at the following path needs to be changed for Luna Client v6.x/7.0:

`"/etc/Chrystoki.conf"`

This configuration file needs to be edited for slot id because by default it is set to 0. Set the slot id to 1 by making the following changes in the configuration file:

```
Presentation = {
OneBaseSlotId = 1;
}
```

Using Luna 6.x/7.0 in FIPS Mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the SafeNet Luna HSM in FIPS mode, you have to make the following change in configuration file:

```
Misc = {
RSAKeyGenMechRemap = 1;
}
```

The above setting redirects the older calling mechanism to a new approved mechanism when SafeNet Luna HSM is in FIPS mode.

NOTE: The above configuration is valid for Luna 7.0 and Luna 6.x (F/W Version 6.22.0 and above only).

2

Integrating SafeNet Luna HSM with BIND Using LunaCA3

Setting up Luna SA with BIND

To perform Luna SA integration with Bind and support for DNSSEC, the DNSSEC toolkit provides the Bind v9.9.2 source which is installed as part of the main build.

Building the source

"dbuild" is a script that builds all source code related to DNSSEC on UNIX platforms.

To build the source, perform the following steps:

1. Traverse to toolkit, e.g. /root/_cdrom_dnssec
2. # make -f dbuild.makefile cleanall
3. # make -f dbuild.makefile clean
4. # make -f dbuild.makefile all VER_LIBXML2=2.7.3

NOTE: For a 32-bit build on Solaris 10 SPARC, run the following:
LUNA_CONFIG_BITS=32 gmake -f dbuild.makefile all VER_LIBXML2=2.7.3

Refer to README-DBUILD under the toolkit to build the source according to your environment.

The toolkit gets installed at:

/opt/SFNTdnssec1

Bind is installed at:

/opt/SFNTdnssec1/bind

Configuring the DNSSEC Toolkit for BIND

1. Traverse to toolkit, e.g. /root/_cdrom_dnssec.
2. Run the OptimizeApache.sh to configure the Luna SA configuration file (/etc/Chrystoki.conf) for BIND.

   ```
   # sh OptimizeApache.sh bind
   ```

   For further information, refer to the README-OPTIMIZE under the DNSSEC toolkit.
3. The Luna SA configuration file (/etc/Chrystoki.conf) is now configured for BIND.

```
Misc = {
Apache = 0;
}
EngineLunaCA3 = {
    EnableDsaGenKeyPair = 1;
    EnableRsaGenKeyPair = 1;
    DisablePublicCrypto = 1;
    EnableRsaSignVerify = 1;
    EnableLoadPubKey = 1;
    EnableLoadPrivKey = 1;
    DisableCheckFinalize = 1;
    DisableEcdsa = 1;
    DisableDsa = 0;
    DisableRand = 0;
    EngineInit = 1:10:11;
    LibPath64 = /usr/lib/libCryptoki2_64.so;
    LibPath = /usr/lib/libCryptoki2.so;
}
```

NOTE: For Solaris SPARC platform in Luna SA v5.0, you need to modify the value of LibPath to /opt/lunasa/lib/libCryptoki2.so in /etc/Chrystoki.conf after running OptimizeApache.sh script.

4. Set the environment by sourcing the **SFNTdnssec.profile** script.

```
# . /opt/SFNTdnssec1/SFNTdnssec.profile
```

5. A sample zone file (foo1.example.net) can be found under the toolkit /<path to toolkit>/_cdrom_dnssec/example. We have used the sample zone file to demonstrate zone-signing using HSM keys.

   Copy the file from the above directory to use

```
$TTL 1d
@ IN SOA foo1.example.net. root.example.net. (
2 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
;
@ IN NS foo1.example.net.
@ IN A 192.168.0.6
foo1 IN A 192.168.0.5
```

```
www IN A 192.168.0.7
gateway IN A 192.168.0.1
```

For more information, refer to README-BIND.

Zone Signing for DNSSEC

We have demonstrated two ways to generate ZSK and KSK to achieve zone signing.

Using “sautil” to generate keys:

Sautil.exe will open the application id once and other applications share login state by setting the same App ID and values. The example below uses App ID values “10:11”
For more information refer “README-BIND” and for configuration refer "README-OPTIMIZE".

1. Open a session to Luna SA using the **sautil** utility:

   ```
   # sautil -v -s 1 -i 10:11 -o –q
   ```

   It will prompt for a token password, provide the Luna SA partition password.

2. Generate a new RSA 2048 keypair (Zone Signing Key):

   ```
   #sautil -v -s 1 -i 10:11 -f Kfoo1zsk.pem -l LABELZSK1 -g 2048
   ```

3. Generate a new RSA 2048 keypair (Key Signing Key):

   ```
   # sautil -v -s 1 -i 10:11 -f Kfoo1ksk.pem -l LABELKSK1 -g 2048
   ```

4. Verify that the two new keypairs are in the HSM:

   ```
   # /opt/SFNTdnssec1/bind/sbin/pkcs11-list -s 1
   ```

5. Import ZSK to dnssec:

   ```
   # /opt/SFNTdnssec1/bind/sbin/dnssec-keyfromlabel -E LunaCA3 -l LABELZSK1 foo1.example.net
   ```

6. Import KSK to dnssec:

   ```
   # /opt/SFNTdnssec1/bind/sbin/dnssec-keyfromlabel -E LunaCA3 -fk -l LABELKSK1 foo1.example.net
   ```

7. Sign the zone file foo1.example.net.

   ```
   # /opt/SFNTdnssec1/bind/sbin/dnssec-signzone -v 9 -E LunaCA3 -S -a foo1.example.net
   ```

8. Close the Opened session:

   ```
   # sautil -v -s 1 -i 10:11 –c
   ```

Using "dnssec-keygen" to generate keys:

Sautil.exe will open the application id once and other applications share login state by setting the same App ID and values. The example below uses App ID values “10:11”

For more information refer “README-BIND” and for configuration refer "README-OPTIMIZE".

1. Open a session to Luna SA using the **sautil** utility:

   ```
   # sautil -v -s 1 -i 10:11 -o –q
   ```

2. Generate a new RSA 2048 keypair (Zone Signing Key):

   ```
   # /opt/SFNTdnssec1/bind/sbin/dnssec-keygen -v 9 -E LunaCA3 -a RSASHA1 -b 2048 foo1.example.net
   ```

3. Generate a new RSA 2048 keypair (Key Signing Key):

```
# /opt/SFNTdnssec1/bind/sbin/dnssec-keygen -v 9 -E LunaCA3 -a RSASHA1 -b 2048 -fk
foo1.example.net
```

4. Verify that the two new keypairs are in the HSM:

```
# /opt/SFNTdnssec1/bind/sbin/pkcs11-list -s 1
```

5. Sign the zone file foo1.example.net.

```
# /opt/SFNTdnssec1/bind/sbin/dnssec-signzone -v 9 -E LunaCA3 -S -a foo1.example.net
```

6. Close the Opened session:

```
# sautil -v -s 1 -i 10:11 -c
```

Bind v9 Deployment demonstrating DNSSEC

The below example demonstrates DNSSEC using a Domain Authority (DA) and a Recursive resolver (RR) server.

NOTE: The Domain Authority and Recursive resolver are already configured without DNSSEC.

Setup Domain Authority server

1. Install and configure Luna SA client on Domain Authority server
2. Check vtl verify.
3. Install the SafeNet DNSSEC toolkit.
4. Salogin to the dns partition.
5. Source environment settings.
6. Flush iptables (iptables - - flush)
7. Start the named server.
8. Using DIG, Verify dns responses from both the DA and the RR.
9. Generate the Zone Signing Key (ZSK) pair.

```
# dnssec-keygen –v 9 -E LunaCA3 –a RSASHA1 –b 1024 testhsm.local
```

10. Generate the Key Signing Key (KSK) pair on the DA.

```
# dnssec-keygen –v 9 –E LunaCA3 –a RSASHA! –b 2048 –fk testhsm.local
```

11. Move the keys to the ../etc directory
12. Sign the zone file.

```
# dnssec-signzone –v 9 –E LunaCA3 –S –a testhsm.local
```

13. Modify the named.conf on the Domain Authority to include:

```
auto-dnssec maintain;

    key-directory "/opt/SFNETdnssec1/bind/etc";
```

14. Using DIG, verify that the RRSIG records are returned from a DIG request both on the DA and the RR servers.

Setup Recursive Resolver server

1. Scp the KeySigningKey .key file to the Recursive Resolver server ../bind/etc directory .
2. Read the key file into the named. conf:

```
trusted-key {
"testhsm.local." 257 3 5 "lfksjalfasdj;……";
};
```

3. Using DIG, verify that the dnssec response is verified by the resolver using the DA public key.

> **NOTE:** Look for the ad (authenticated data) flag. The "ad" flag, or Authenticated Data, indicates that the signatures validated correctly. You are now assured that the response you have received is honestly and truly from the official source. You can also check for DO (DNSSEC OK) flag in EDNS section.

4. Modify the public key and verify that the dns request fails (SERVFAIL response).

> **NOTE:** Please refer the ISC BIND Administrators Reference Manual for more details.

3

Integrating SafeNet Luna HSM with OpenDNSSEC Using Luna CA3

To perform Luna SA integration with OpenDNSSEC, the DNSSEC toolkit provides the OpenDNSSEC v1.3.12 source which is installed as part of the main build.

Building the source

"dbuild" is a script that builds all source code related to DNSSEC on UNIX platforms.

To build the source, perform the following steps:

1. Traverse to toolkit, e.g. /root/_cdrom_dnssec
2. make -f dbuild.makefile cleanall
3. make -f dbuild.makefile clean
4. make -f dbuild.makefile all VER_LIBXML2=2.7.3

Refer to README-DBUILD under the toolkit to build the source according to your environment.

The toolkit gets installed at:

/opt/SFNTdnssec1

OpenDNSSEC is installed at:

/opt/SFNTdnssec1/bind

Configuring the DNSSEC Toolkit for OpenDNSSEC

To setup the DNSSEC toolkit for BIND, follow the steps below:

1. Traverse to toolkit, e.g. /root/_cdrom_dnssec.
2. Run the OptimizeApache.sh to configure the Luna SA configuration file (/etc/Chrystoki.conf) for OpenDNSSEC.

   ```
   # sh OptimizeApache.sh opendnssec
   ```

   For further information, refer to the README-OPTIMIZE under the DNSSEC toolkit.
3. The Luna SA configuration file (/etc/Chrystoki.conf) is now configured for OpenDNSSEC.

   ```
   Misc = {

   Apache = 0;
   }
   ```

```
EngineLunaCA3 = {
EnableDsaGenKeyPair = 1;
EnableRsaGenKeyPair = 1;
DisablePublicCrypto = 1;
EnableRsaSignVerify = 1;
EnableLoadPubKey = 1;
EnableLoadPrivKey = 1;
DisableCheckFinalize = 1;
DisableEcdsa = 1;
DisableDsa = 0;
DisableRand = 0;
EngineInit = 1:10:11;
LibPath64 = /usr/lib/libCryptoki2_64.so;
LibPath = /usr/lib/libCryptoki2.so;
}
```

4. Set the environment by sourcing the **SFNTdnssec.profile** script.

```
# . /opt/SFNTdnssec1/SFNTdnssec.profile
```

5. Edit 'Repository name', 'Module', 'TokenLabel', and 'PIN' in the following file:

```
"/opt/SFNTdnssec1/opendnssec/etc/opendnssec/conf.xml"

e.g.,

<Repository name="part1">

<Module>/usr/lib/libCryptoki2.so</Module>

<TokenLabel>part1</TokenLabel>

<PIN>temp123#</PIN>

</Repository>

where:

part1 -> HSM partition

/usr/lib/libCryptoki2.so -> PKCS#11 library

temp123# -> HSM partition PIN
```

6. Edit 'Repository' in the kasp.xml file

```
"/opt/SFNTdnssec1/opendnssec/etc/opendnssec/kasp.xml"

        e.g., storing KSK and ZSK in the same partition:

        <!-- Parameters for KSK only -->

        <KSK>

        <Algorithm length="2048">7</Algorithm>

        <Lifetime>P1Y</Lifetime>

        <Repository>part1</Repository>

        <Standby>1</Standby>

        <!-- <ManualRollover/> -->

        </KSK>

        <!-- Parameters for ZSK only -->

        <ZSK>

        <Algorithm length="1024">7</Algorithm>
```

```
<Lifetime>P30D</Lifetime>
<Repository>part1</Repository>
<Standby>1</Standby>
</ZSK>
```

7. Create a sample zone file (/var/ foo1.example.net).

```
$TTL 1d
@ IN SOA foo1.example.net. root.example.net. (
2 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
;
@ IN NS foo1.example.net.
@ IN A 192.168.0.6
foo1 IN A 192.168.0.5
www IN A 192.168.0.7
gateway IN A 192.168.0.1
```

For more information, refer to README-OPENDNSSEC.

Creating the repository for OpenDNSSEC and test HSM

1. List existing repositories (if any).

```
# ods-control hsm list
```

2. Create the repository ("part1").

```
# ods-control hsm generate part1 rsa 1024
```

3. List the repository.

```
# ods-control hsm list part1
```

4. Test the repository (optional).

```
# ods-control hsm test part1
```

5. Test signing performance (optional).

```
# ods-hsmspeed -r part1 -i 2 -s 1024 -t 15
```

Setup keys for OpenDNSSEC

1. Prepare opendnssec for use.

```
# ods-control ksm update all
```

NOTE: On issuing the above command, an error is thrown as show below:

SQLite database set to: /opt/SFNTdnssec1/opendnssec/var/opendnssec/kasp.db

File /opt/SFNTdnssec1/opendnssec/var/opendnssec/kasp.db does not exist, nothing to backup

ERROR: error executing SQL - no such table: dbadmin

Failed to connect to database.

The workaround is to run the below command:

```
# /opt/SFNTdnssec1/sqlite/bin/sqlite3
/opt/SFNTdnssec1/opendnssec/var/opendnssec/kasp.db <
/root/_cdrom_dnssec/opendnssec-1.3.12/enforcer/utils/database_create.sqlite3
```

2. List repository.

```
# ods-control ksm repository list
```

3. Add the zone.

```
#ontrol ksm zone add -z foo1.example.net -p default -i /var/foo1.example.net -o
/var/foo1.example.net.signed
```

4. List the zone.

```
# ods-control ksm zone list
```

5. List the keys.

```
# ods-control ksm key list --zone foo1.example.net
```

NOTE: Remember to start the signer daemon (i.e., command "ods-control start").

4

Integrating SafeNet Luna HSM with OpenDNSSEC and BIND Using GemEngine

Before You Begin

It is recommended that you should familiarize yourself with OpenSSL. Refer to the appropriate documents for OpenSSL commands at the following location:

http://www.openssl.org/docs/

OpenSSL provides the support of engine (basically hardware devices) to store the keys on hardware devices to make keys more secure. SafeNet provides the OpenSSL toolkit having support of GEM Engine that is used to communicate with the SafeNet Luna HSM.

OpenSSL Toolkit

The OpenSSL toolkit is provided to make the installation quick and easy. The installation CD can be obtained from the SafeNet Customer Support.

NOTE: For GemEngine v1.2 setup, contact Customer support.

Doc IDs for GemEngine v1.2 is DOW0002177/KB0016309.

OpenSSL Configuration

Configure OpenSSL to enable Gem Engine by default.

It is assumed that OpenSSL with Luna engine is installed at the location /usr/local/ssl/bin/openssl

1. Locate the OpenSSL engines directory using gembuild. It is available in the gem engine directory that is downloaded and extracted.

   ```
   # ./gembuild locate-engines
   ```

   The openssl engines directory is located at `/usr/local/lib64/engines.`

   This displays the engines directory for the OpenSSL that is in PATH.

2. Copy the libgem.so to the engines directory and test engine.

   Example:

   ```
   # cp builds/linux/rhel/64/1.0.1/libgem.so /usr/local/lib64/engines/
   ```

3. Verify the gem engine is present.

```
# openssl engine gem -v
     (gem) Gem engine support
     enginearg, openSession, closeSession, login, logout, engineinit,
     CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
     SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE,
     IntermediateProcesses
```

   If the output looks as above then the gem engine is successfully installed.

4. Locate the openssl.cnf and engines directory.

```
# openssl version -d
OPENSSLDIR: "/usr/local/ssl" or OPENSSLDIR: "/etc/pki/tls" default
```

   The openssl.cnf file is located in the above directory. Note that this is for the OpenSSL that is in PATH. Run "which openssl" to ensure it is the right one.

```
# ./gembuild locate-engines
```

   This gives the directory where the libgem.so should be located. Note that this is for the OpenSSL that is in PATH.

5. Edit the openssl.cnf file.

   Example:

```
# Insert near top of file openssl.cnf:
    openssl_conf = openssl_init
    # Insert at bottom of file openssl.cnf:
    [ openssl_init ]
    engines = engine_section
    [ engine_section ]
    gem = gem_section
    [ gem_section ]
    dynamic_path = /usr/local/ssl/lib/engines/libgem.so
    default_algorithms = ALL
```

6. Verify that engine is loading without specifying it.

```
# openssl engine -v
(dynamic) Dynamic engine loading support
        SO_PATH, NO_VCHECK, ID, LIST_ADD, DIR_LOAD, DIR_ADD, LOAD
    (gem) Gem engine support
         enginearg, openSession, closeSession, login, logout, engineinit,
         CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
         SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE,
```

7. Test the application. First generate the key and then generate certificate request.

```
# /usr/local/ssl/bin/openssl genrsa 2048
```

openssl command-line WITHOUT explicit engine:

```
# openssl req -new -nodes -key tmpkey.pem -out tmpkey.req -days 30 –verify
```

Where tmpkey.pem is output generated by “openssl genrsa” command. It will use the Gem engine by default without mentioning it in command line.

Install and Configure BIND and DNSSEC

1. Download and extract the source for bind and opendnssec. The dbuild.inc under gemengine/dnssec folder refers to required software packages.Place the .tar.gz files inside of the gemengine/dnssec directory.
2. Run gembuild to build and install bind

   #./gembuild dnssec-makebind

   NOTE: To build and install opendnssec and bind, please run below command.

   #./gembuild dnssec-makeall

3. Set the environment by sourcing the **SFNTdnssec.profile** script.

   ```
   # . /opt/SFNTdnssec1/SFNTdnssec.profile
   ```

4. Create a sample zone file to sign named as "`foo1.example.net`" in "`/opt/SFNTdnssec1/`" directory.
5. Open "`foo1.example.net`" file and enter below entry:

   ```
   $TTL 1d
   @ IN SOA foo1.example.net. root.example.net. (
           2 ; Serial
           604800 ; Refresh
           86400 ; Retry
           2419200 ; Expire
           604800 ) ; Negative Cache TTL
   ;
   @ IN NS foo1.example.net.
   @ IN A 192.168.0.6
   foo1 IN A 192.168.0.5
   www IN A 192.168.0.7
   gateway IN A 192.168.0.1
   ```

For more information, refer to README-BIND

Zone Signing for DNSSEC

We have demonstrated two ways to generate ZSK and KSK to achieve zone signing.

Using “sautil” to generate keys

Sautil.exe will open the application id once and other applications share login state by setting the same App ID and values. The example below uses App ID values “10:11”
For more information refer “README-BIND” and for configuration refer "README-OPTIMIZE".

1. Open a session to Luna SA using the **sautil** utility:

   ```
   # sautil -v -s 1 -i 10:11 -o –q
   ```

2. Generate a new RSA 2048 keypair (Zone Signing Key):

   ```
   #sautil -v -s 1 -i 10:11 -f Kfoo1zsk.pem -l LABELZSK1 -g 2048
   ```

3. Generate a new RSA 2048 keypair (Key Signing Key):

   ```
   # sautil -v -s 1 -i 10:11 -f Kfoo1ksk.pem -l LABELKSK1 -g 2048
   ```

4. Verify that the two new keypairs are in the HSM:

   ```
   # /opt/SFNTdnssec1/bind/sbin/pkcs11-list -s 1
   ```

5. Import ZSK to dnssec:

   ```
   # dnssec-keyfromlabel -E gem -l LABELZSK1 foo1.example.net
   ```

6. Import KSK to dnssec:

   ```
   # dnssec-keyfromlabel -E gem -fk -l LABELKSK1 foo1.example.net
   ```

7. Sign the zone file foo1.example.net.

   ```
   # dnssec-signzone -v 9 -E gem -S -a foo1.example.net
   ```

8. Close the Opened session:

   ```
   # sautil -v -s 1 -i 10:11 –c
   ```

Using "dnssec-keygen" to generate keys

Sautil.exe will open the application id once and other applications share login state by setting the same App ID and values. The example below uses App ID values “10:11”
For more information refer “README-BIND” and for configuration refer "README-OPTIMIZE".

1. Open a session to Luna SA using the **sautil** utility:

   ```
   # sautil -v -s 1 -i 10:11 -o –q
   ```

2. Generate a new RSA 2048 keypair (Zone Signing Key):

   ```
   # dnssec-keygen -v 9 -E gem -a NSEC3RSASHA1 -b 2048 foo1.example.net
   ```

3. Generate a new RSA 2048 keypair (Key Signing Key):

   ```
   # dnssec-keygen -v 9 -E gem -a NSEC3RSASHA1 -b 2048 -fk foo1.example.net
   ```

4. Verify that the two new keypairs are in the HSM:

   ```
   # /opt/SFNTdnssec1/bind/sbin/pkcs11-list -s 1
   ```

5. Sign the zone file foo1.example.net.

   ```
   # dnssec-signzone -v 9 -E gem -S -a foo1.example.net
   ```

6. Close the session:

   ```
   # sautil -v -s 1 -i 10:11 -c
   ```