DNS Security Extensions (DNSSEC)

Integration Guide

SafeNet. THE DATA PROTECTION COMPANY

Part Number: 007-011195-001 (Rev B, 03/2013)

Preface

SafeNet invites constructive comments on the contents of this document. These comments, together with your personal and/or company details, should be sent to the address below.

4690 Millennium Drive
Belcamp, Maryland 21017, USA

Limitations

This document does not include the steps to set up the third-party software. The steps given in this document must be modified accordingly. Refer to Luna SA documentation for general Luna setup procedures.

Disclaimers

The foregoing integration was performed and tested only with the specific versions of equipment and software and only in the configuration indicated. If your setup matches exactly, you should expect no trouble, and Customer Support can assist with any missteps. If your setup differs, then the foregoing is merely a template and you will need to adjust the instructions to fit your situation. Customer Support will attempt to assist, but cannot guarantee success in setups that we have not tested.

Technical Support

If you encounter a problem while installing, registering or operating this product, please make sure that you have read the documentation. If you cannot resolve the issue, please contact your supplier or SafeNet support.

SafeNet support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between SafeNet and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Technical Support Contact Information:

Phone: 1-800-545-6608, 1-410-931-7520
Email: support@safenet-inc.com

Table of Contents

Chapter 1
Introduction

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 SA Hardware Security Module (HSM).

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

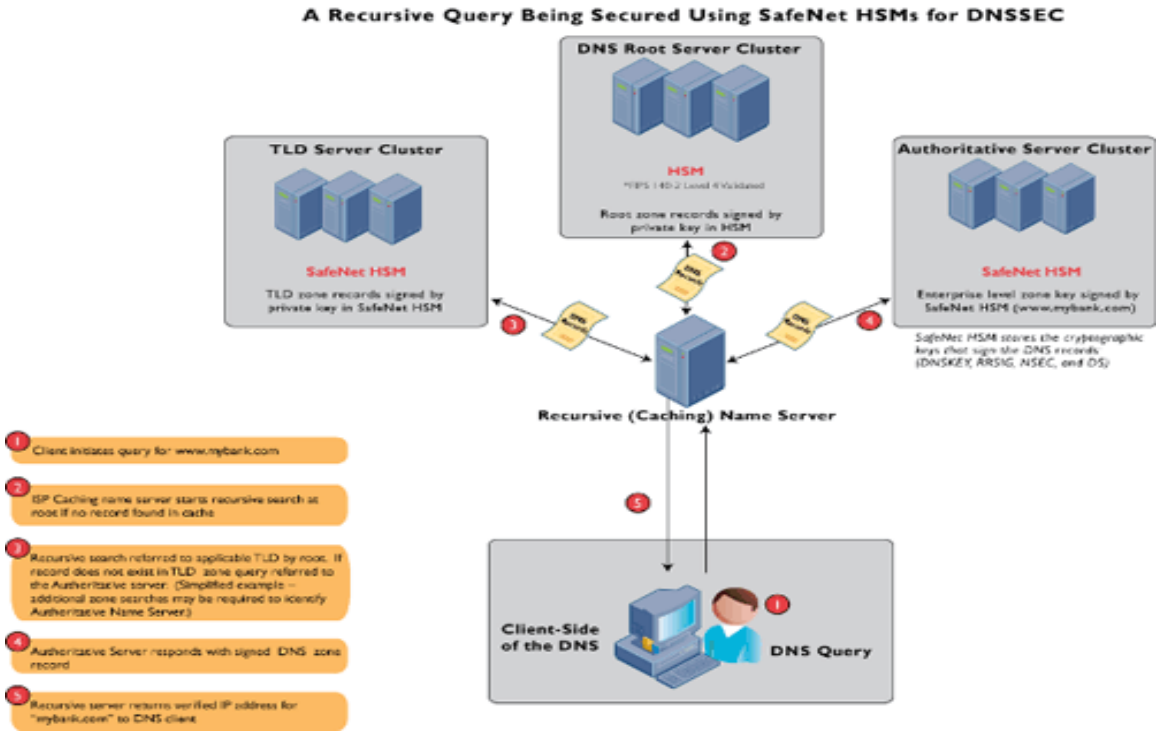
What is DNSSEC:

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.



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.

3rd Party Application Details

- BIND (v9.9.2/v9.7)
- Open DNSSEC (v1.3.12/v1.0.0)

Supported Platforms

The following platforms are supported for Luna SA v5.1

- Debian Linux 6.0 (64 bit)

The following platforms are supported for Luna SA v5.0:

- RHEL 5.4 (32-bit / 64-bit)
- Solaris 10 SPARC (32-bit)

HSM and Firmware Support

We did this integration with the following:

- Luna SA v5.1 f/w 6.2.1 with Luna SA Client s/w v5.1 (64-bit)
- Luna SA v5.0 f/w 6.0.7 with Luna SA Client s/w v5.0 (32-bit/64-bit)
- DNSSEC Toolkit

Prerequisites

Luna SA Setup

Please refer to the Luna SA documentation for installation steps and details regarding configuring and setting up the box on Windows systems. Before you get started ensure the following:

- Luna SA appliance and a secure admin password
- Luna SA, and a hostname, suitable for your network
- Luna SA network parameters are set to work with your network
- Initialized the HSM on the Luna SA appliance.
- Create and exchange certificates between the Luna SA 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 Luna SA. The general form of command is C:\Program Files\Luna SA > vtl verify for Windows.
- Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to Luna SA with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

DNSSEC Toolkit

The DNSSEC toolkit is provided to make the installation quick and easy. The installation CD can be obtained from the SafeNet Customer Connection Center.

Chapter 2
Integrating BIND with Luna SA

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**

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

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

a) Traverse to toolkit, e.g. /root/_cdrom_dnssec.

b) 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.

c) 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;
}
```

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.

d) Set the environment by sourcing the **SFNTdnssec.profile** script.

 # . /opt/SFNTdnssec1/SFNTdnssec.profile

e) 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.

1. 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”

a) Open a session to Luna SA using the **sautil** utility provided under **/opt/SFNTdnssec1/sautil/bin**:

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

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

b) Generate a new RSA 1024 bit keypair which is the Zone Signing Key (ZSK) using sautil.

 # sautil -v -s 1 -i 10:11 -f Kfoo1zsk.pem -l LABELZSK1 -g 1024

c) Generate a new RSA 2048 bit keypair which is the Key Signing Key (KSK) using sautil.

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

d) Check whether the new keypair are on the HSM:

 # pkcs11-list -s 1

e) Import the Zone Signing Key (ZSK) to DNSSEC.

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

f) Import the Key Signing Key to DNSSEC.

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

g) Sign the zone file foo1.example.net.

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

h) Close the session.

 # sautil -v -s 1 -i 10:11 –c

2. Using “dnssec-keygen” to generate keys:

This is the direct way to generate keys for BIND/DNSSEC. Refer “README-OPTIMIZE” for configuration options.

a) Open a session to Luna SA using the **sautil** utility provided under **/opt/SFNTdnssec1/sautil/bin**.

 # sautil -v -s 1 -i 10:11 -o –q

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

b) Generate a new Zone Signing Key:

 # /opt/SFNTdnssec1/bind/sbin/dnssec-keygen -v 9 -E LunaCA3 -a RSASHA1 -b 1024 foo1.example.net

c) Generate a new Key Signing Key:

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

d) Check whether the new keypair are on the HSM.

 # pkcs11-list -s 1

e) Sign the zone file foo1.example.net.

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

f) Close the 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.

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

Setup Domain Authority server

a) Install and configure Luna SA client on Domain Authority server

b) Check vtl verify.

c) Install the SafeNet DNSSEC toolkit.

d) Salogin to the dns partition.

e) Source environment settings.

f) Flush iptables (iptables - - flush)

g) Start the named server.

h) Using DIG, Verify dns responses from both the DA and the RR.

i) Generate the Zone Signing Key (ZSK) pair.

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

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

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

k) Move the keys to the ../etc directory

l) Sign the zone file.

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

m) Modify the named.conf on the Domain Authority to include:

auto-dnssec maintain;

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

n) 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

a) Scp the KeySigningKey .key file to the Recursive Resolver server ../bind/etc directory

b) Read the key file into the named. conf:

trusted-key {

"testhsm.local." 257 3 5 "lfksjalfasdj;……";

};

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

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.

d) 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.

Chapter 3
Integrating OpenDNSSEC with Luna SA

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:

a) Traverse to toolkit, e.g. /root/_cdrom_dnssec.

b) 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.

c) 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;
}
```

d) Set the environment by sourcing the **SFNTdnssec.profile** script.

. /opt/SFNTdnssec1/SFNTdnssec.profile

e) 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

f) 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>
```

g) 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

a) List existing repositories (if any).

 # ods-control hsm list

b) Create the repository ("part1").

 # ods-control hsm generate part1 rsa 1024

c) List the repository.

 # ods-control hsm list part1

d) Test the repository (optional).

 # ods-control hsm test part1

e) Test signing performance (optional).

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

Setup keys for OpenDNSSEC

a) Prepare opendnssec for use.

 # ods-control ksm update all

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 command below:

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

b) List repository.

 # ods-control ksm repository list

c) Add the zone.

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

d) List the zone.

 # ods-control ksm zone list

e) List the keys.

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

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