Secure Routing Extensions Crypto API

– SRxCryptoAPI –

Version 0.2.0

User's Manual

July 2017

http://bgpsrx.antd.nist.gov

bgpsrx-dev@nist.gov

Oliver Borchert, Kyehwan Lee, Kotikalapudi Sriram, Doug Montgomery

Index

Introduction

Overview

SRxCryptoAPI is a wrapper library that manages Cryptography plugins for BGPsec path processing. This library allows embedding configuring BGPsec crypto implementations into software packages such as QuaggaSRx and SRX-Server. Furthermore, the plug and play design allows to switch out cryptography implementations without the need of recompiling neither QuaggaSRx nor SRx-Server. Both, QuaggaSRx and SRx-Server do require this API.

The SRxCryptoAPI library (SCA) is shipped with an OpenSSL based implementation for BGPsec path validation and signing called BGPSecOpeSSL. In addition, SCA provides a test implementation for debugging purpose called SRxCryptoTestlib.

It is possible to write other implementations and link them using the configuration file.

Furthermore, the SRxCryptoAPI software package provides a set of tools which allow the generation of keys and certificates used by QuaggaSRx / SRx-Server.

Installation

For Red Hat based systems the software can be installed using the 'yum' installer. The repository information is available on the website http://bgpsrx.antd.nist.gov

For source-based installation read the INSTALL file provided.

SRxCryptoAPI configuration

The API requires a configuration script that is located either in the installation '*/etc/*' directory or in '*./*'. The configuration has the following format:

library_conf = "plugin-config-name";

Specifies the configuration to be used. SRxCryptoAPI allows configuring multiple plugins within the main configuration file. This setting selects the configuration that must be loaded.

key_volt = "directory for the keys to be stored";

Specifies the location where the public and private keys are stored. This setting can be set programmatically. It is used by the key helper methods that are part of the wrapper API, not the plugin API.

key_ext_private = "der";

Specifies the filename extension used by files containing the private key in DER format.

key_ext_public = "cert";

Specifies the filename extension used by X509 certificate files containing the public key in DER format.

Debug-type = <int>;

Specifies the debugging type. Only information that matches the debugging type or are less in its numerical value are displayed. Plugins might not allow setting the debug type. The value can range from 0-7 [0:Emergency,1:Alert,2:Critical,3:Error, 4:Warning, 5:Notice, 6:Info, 7:Debug].

Plugin Configuration Block

Each plugin is configured in a plugin configuration block. Each block is scripted in the following format:

```
plugin-config-name: {
  …
}
```

Within each plugin configuration block is the configuration of the plugin. The library name is required, all other method mappings are needed only in case the plugin uses different function names as proposed by the API itself.

library_name = "custom-library.so"

Specifies the library that will be loaded and wrapped by the SRxCryptoAPI. This parameter is mandatory. Two libraries are provided: *libSRxBGPSecOpenSSL.so* and *libSrxCryptoTestLib.so*

The following configuration settings specify the method mappings in case the plug in uses different method names. In the listing below we map to the default method names. For detailed API function description see the ***srxcryptoapi.h*** header file.

To determine the extent of functionality the plugin provides, consult the documentation provided with the plugin. The minimum requirement a plugin must fulfill are the two methods **validate** and **sign**, whereas the method names can differ. In case the names differ from the default names specified in the srxcryptoapi.h header file, a configuration for the method mapping as displayed below must be provided.

init_value = "TEXT|NULL"

This configuration setting allows the generation of an initialization string. Consult the documentation of the crypto plugin regarding the syntax of the initialization value. Both plugins provided with SCA allow the specification of an initialization string. Use the value "NULL" in case no initialization string is provided.

BGPSecOpenSSL: "BUP:<filename>;PRIV:<filename;" or "NULL"
Filename: name of the file containing a list of ASN, Key SKI's in the format: <ASN>-SKI: <SKI> (one per line)

SRxCryptoTestLib: "0|1;0|1;0|1" or "NULL"
1. Parameter: Specifies the result for validation (0 valid, 1 invalid)
2. Parameter: Indicates if the provided BGPsec_PATH attribute should be processed.
3. Parameter: Indicates if the found signatures should be printed out. (requires "1" as second parameter)

method_init = "init"

Will be called during initialization. The *init* method has three parameters, a string, the debug level, and a status flag. The status flag is an out parameter. The content of the string depends on the plugin which is configured – It allows to pass an initialization command or other to the module.

method_validate = "release"

Release will be called before the library will be unloaded. It provides one status flag which is an OUT parameter.

method_validate = "validate"

Validate will be called for validation

method_sign = "sign"

Sign the given path with the private key provided.

method_freeHashMessage = " freeHashMessage"

The hash message can be the output from the validation command. This is the message the crypto module uses to calculate the hash from. In case the API returns the hash message, the caller is required to call freeHashMessage on the API to free the allocated memeory.

method_freeSignature = " freeSignature"

Same as with the hash message, the API generates the memory used for the signature. The API must be used to de-allocate the memory again.

Key Management

To allow BGPsec path validation and BGPsec path signing, all required keys must be registered. Only registered keys are used for validation and signing.

method_registerPrivateKey = "registerPrivateKey"

Register the given private key. (private key management must be enabled)

method_unregisterPrivateKey = "unregisterPrivateKey"

Unregister the previously registered key. (private key management must be enabled)

method_registerPublicKey = "registerPublicKey"

The plugin provides its own key storage and allows storing the public key provided. (public key management must be enabled)

method_unregisterPublicKey = "unregisterPublicKey"

Remove the previously stored public key from the key storage. (public key management must be enabled)

Debug functions

The Debug functions allow to control the debug level of the module. The module does not need to provide additional debugging information but it helps in case something does not work as expected.

method_getDebugLevel = " getDebugLevel"

Retrieve the current debug level or -1 if not supported.

method_setDebugLevel = " setDebugLevel"

Retrieve the previous debug level or -1 if not supported.

Key Generation Tools

The tools provided with the SRxCryptoAPI allow to generate private and public keys for test purpose only! Keys should normally be received by a validation cache or through other means.

To setup a "key_volt" for SRxCryptoAPI perform the following steps:

1. Generate a repository e.g. /var/lib/bgpsec-keys
   This repository will contain the keys and certificates generated by the key generation tools. In this version, the certificates only contain the key.
2. Generate the Keys:
   qsrx-make-key <name>
3. Generate the Certificate
   qsrx_make-cert <name>
4. Publish the cert and install the Key. In addition, this step generates a file containing all SKI's for private keys for easy copy and paste into the router configuration.
   qsrx-publish <key>

The latest version also allows to generate self-signed public key certificates containing all information specified in RFC8208. For this use the script **qsrx-make-router-cert.sh**

All tools allow the parameter **-?** to inquire more help on usage and options.

Support

In case of crashes, please provide a description on how to reproduce the crash and if possible a core dump.

To be informed of bug fixes or ask questions to the community, subscribe to the users email list by sending an email to bgpsrx-users-request@nist.gov with subscribe in the subject.

Questions to the developers and general contact information:

Email: bgpsrx-dev@nist.gov
Web: https://bgpsrx.antd.nist.gov

Developers:

Oliver Borchert oliver.borchert@nist.gov
Kyehwan Lee