BGP Secure Routing Extension

– BGP SRx-Server –

Version 0.5.0

User's Manual

July 2017

www-x.antd.nist.gov/bgpsrx

bgpsrx-dev@nist.gov

Oliver Borchert, Kyehwan Lee, Kotikalapudi Sriram, Doug Montgomery

Index

Introduction

The BGP SRx-Server is developed by the Advanced Network Technology Division (ANTD) at the National Institute of Standards and Technology (NIST). This effort is supported by the Department of Homeland Security under the Secure Protocols for the Routing Infrastructure (SPRI) program and the NIST Information Technology Laboratory Cyber and Network Security Program.

The SRx-Server provides an API to router implementations that wish to perform BGP origin validation and BGPsec path validation. The SRx-Server is meant to be used as a validation service. This keeps the impact (software/memory/processing) on the router to a minimum. SRx-Server provides an API that allows embedding a small proxy into the router which in return communicates to the SRx-Server. This design allows the SRx-Server to serve multiple BGP router instances.

What BGP SRx-Server is not: BGP SRx-Server is not an RPKI validation cache. It does not rsync with RPKI repositories nor does is validate certificates. The SRx-Server synchronizes with RPKI validation caches using the RPKI to router protocol as described in RFC 6810/8210. At this point, the SRx-server uses plain TCP as transport for communication with the RPKI cache(s).

Status of BGP SRx-Server

The current version of BGP SRx-Server is NOT intended to be used in a production system, even though we attempt to get it to production level. BGP SRx-Server is in the stage of a prototype / reference implementation and might face some instability. In such a case, we appreciate every input that helps us to improve the stability as well as performance. The code is open source. Feedback can be send to bgpsrx-dev@nist.gov.

Important Changes:

The SRx-Server updated the router to cache protocol from RFC 6810 to RFC 8210 which includes the transmission of router keys. Also, the SRx-Server added BGPsec path validation as specified in RFC 8205. It requires the SRxCryptoAPI for all crypto operations.

Installation

Please see the SRxSoftwareSuite-5.0-QuickInstallGuide for the easy install.

BGP SRx-Server provides configuration scripts. To install SRx, download the package from http://www-x.antd.nist.gov/bgpsrx. Once downloaded, deflate the package and call the configuration method. The file labeled "INSTALL" contains an example on how to configure the SRx server. This package also generates the SRx-

API that is needed for QuaggaSRx. Follow the configuration messages and install missing libraries as needed. The internal prefix tree (Patricia) tree is bundled in this package. If you decide to use the default one, check the patch file located in SRx/extras and update the installed version. Recompilation will be necessary.
If not, use the switch --with-patr, and SRx-Server will be compiled with the bundled version.
Once compiled, call "make; make install". To select an individual installation directory, use the configuration parameter --prefix. This directory is needed for QuaggaSRx to specify the location of the API binaries.

Note: The software is developed and tested primarily on CentOS 6 and Centos 7

The SRx Server Software [srx_server]

The BGP-SRx Server is the main component that collects ROA information from the RPKI validation cache using the router to cache protocol (RFC 6810/8210). The BGP router connects to SRx using the SRx-API and sends update validation requests for origin validation and path validation. The SRx server answers by sending the validation results either in two separate notifications, one for origin validation and one for path validation depending on availability or bundled in one single notification containing both results. The validation results are processed independently from each other as pure origin validation and pure path validation. The path validation process does NOT include origin validation. The combining of these two results for the BGP decision process MUST be done within the router. This allows for implementing different customized strategies in the router and is very helpful for research purposes.
Once the SRx Server recognizes a change in the validation state of either of the validations, it sends the appropriate notification to all routers, registered for the update.
The SRx Server can be remotely accessed by using a telnet client. It is recommended to combine telnet with rlwrap to gain command history. Using the alias command allows easy combination of rlwrap with telnet: alias telnet='rlwrap telnet'.
As soon as a telnet session is established (only a single session is supported at this time) a list of all available commands can be retrieved using the *'help'* command.

Available commands are:

close, quit, exit	Close this console!
shutdown <password>	Shutdown the SRx Server!
log-level [<number>]	Set or show the log level of the server. If no log level is provided, the server returns the current log level. The following log-levels are supported: 3=ERROR, 4=WARNING, 5=NOTICE, 6=INFO, 7=DEBUG
rtr-sync [proxyID]	Send synchronization request to the provided proxy or all. (Currently to all)

num-updates	Display the number of updates stored in the update cache and shadows stored in the prefix cache. Only updates in the prefix cache are verified.
num-prefixes	Display the number of prefixes stored in the prefix cache!
num-proxies	Display the number of proxies attached.
command-queue	Displays the number of queued commands in the command queue.

show-srxconfig	Display the configuration of the SRx server
show-update <id>	Display update data with the ID (hex or decimal).
show-proxies	Display the proxy mapping.

dump-ucache	Dump the update cache to command line of SRx!
dump-pcache	Dump the prefix cache to command line of SRx!

(WARNING: the dump-xxxx commands dump the complete cache content on the command line. This function should only be used for debugging of small data sets!)

!! [<parameter>]	Repeat last command with optional new parameter if specified, otherwise old parameter!

SRx Configuration Settings:

The SRx Server requires a configuration file which is expected to be either in the directory from where it is called or provided using the parameter –f <conf-file>. Most configuration settings can be handed over as command line parameters. In case of a conflict to between a command line parameter and the corresponding configuration script, the command line parameter has precedence over the configuration script. The server will NOT start without a configuration file.

Configuration File Parameters:

```
# print information on the command console
# verbose  = true|false;
verbose = true

# specifies the log level for output (3=ERROR, 4=WRNING, 5=NOTICE,
# 6=INFO, 7=DEBUG)
#loglevel = 3|4|5|6|7;
loglevel = 3;

# specify the log file name, otherwise log information will be send to the
# console.
log    = "/var/log/srx_server.log";
```

```
# if enabled the SRx server will send a synchronization request to the router.
# It is expected that the router will send validation requests for all updates in
# its tables to SRx server.
# sync = true|false;
sync    = true;

# The port address the SRx server is listening on for connections from the
# router.
port = 17900;

# Console configuration:
console: {
        # Port address of the server console.
        port = 17901;
        # The password used to shutdown the BGP-SRx server
        password = "x";
};

# Configuration for Validation Cache:
rpki: {
        # Server address of the validation cache
        host = "localhost";
        # Port address of the validation cache. Protocol: router to cache
        port = 323;
        # supports 2 versions: 0 => RFC6810, 1=> RFC8210
};

# Configuration for BGPsec integration
bgpsec: {
        # Allows to set the SCA configuration file for path validation
        #srxcryptoapi_cfg="<configuration file>"
        srxcryptoapi_cfg="/etc/srxcryptoapi.cfg"

        # Synchronize the logging settings of SCA with the logging settings of
        # srx-server. If set to false, the SCA configuration takes precedence
        sync_logging = true;
};

# These experimental settings are used to manipulate the internal execution
# flow of the server.
mode: {
        # Turn off the send queue (true|false)
        no-sendqueue = true;
        # turns off the receiver queue (true|false)
        no-receivequeue = false;
};
```

```
# It is possible to pre-configure the internal proxy-client mapping. This
# mapping is used to identify which client is linked to what update. It is
# possible to configure up to max 255 clients.
mapping: {
        # client_x = y  with x = 1..255 and y either an IPv4 or 4-byte integer
        client_1        = "2";
        client_10       = "10.0.0.1";
};
```

IMPORTANT: Command line parameters overrule configuration script parameters.

Configuration as command line parameter

-h, --help	Display this help and exit
-f <file>, --file <file>	Specify a configuration file
--credits	Displays the developers information
--version	Displays the version number
--full-version	Displays the full version number
-v, --verbose	Enable verbose output
--loglevel <level>	The log level for the verbose output. The following levels are supported: (3)=ERROR, (4)=WANRING, (5)=NOTICE, (6)=INFO, (7)=DEBUG
-l <file>, --log <file>	Write all messages to a file
--syslog	Send all messages to syslog
-C <#>, --proxy-clients <#>	Minimum expected number of proxy clients. By default, this value is 2. It affects the internal memory consumption / performance per update.
-s --sync	Send synchronization request each time a proxy connection is established!
-k --keep-window <sec>	The default keepWindow in seconds. Zero deactivates this feature.
-p, --port <#>	Specify the listening port (def.: 17900)
-c, --console.port <#>	Specify the console port (def.: 17901)
-P <pwd>, --console.password <pwd>	Password for remote shutdown
--rpki.host <name/ip>	RPKI/Router protocol server host name
--rpki.port <#>	RPKI/Router protocol server port number
--bgpsec.host <name/ip>	BGPsec/Router protocol server host name
--bgpsec.port <#>	BGPsec/Router protocol server port number
--mapping.client_# <ip/#>	A pre-defined proxy/client mapping. The client number ranges from 1..255 the proxy id can be given as either IPv4 or 4-byte integer

Experimental Options:

--mode.no-sendqueue	Disable send queue for immediate results.
--mode.no-receivequeue	Disable the receive queue. This queue allows to push the processing of packets into its own thread.

Tools

This software package comes with a set of tools used to test SRx-server and its components. These tools simulate a validation cache, a validation cache client, and a router / SRx-client.

Server Tools

Validation Cache Test Harness: rpkirtr_svr

command: prkirtr_svr [port – Def: 323]

The tool is a Validation Cache Test Harness / Simulator. The default port is 323. It provides the validation cache interface to

The rpkirtr_svr and can be used to inject ROA information as well as public keys to the system. The command line console can be used to add and delete ROA/Key entries. In addition, it allows loading ROA/Key information via a script. Refer to the example files provided in this distribution.

This tool provides command auto completion and a more detailed Help command.

Console Commands

verbose
: Toggle to turn verbose mode on and off

cache
: Display the content of the validation cache

version
: Display the version of the tool.

sessionID
: Display the current session id

help [<command>]
: Display this screen or detailed help for the given command!

credits
: Display credits information!

empty
: Empties the cache

sessionID <number>
: Generates a new session id.

append <filename>
: Appends a prefix file's content to the cache

add <prefix> <maxlen> <as>
: Manually add a ROA whitelist entry

addNow <prefix> <maxlen> <as>
: Manually add a ROA whitelist entry without any delay!

keyLoc <directory>
: Configure a directory location of keys that will be pre-pended to every certificate-file.

addKey <asn> <certificate-file>
: Manually add a router key whitelist entry

addKeyNow <asn> <certificate-file>
Manually add a router key whitelist entry without any delay!
remove <index> [end-index]
Remove one or more cache entries
removeNow <index> [end-index]
Remove one or more cache entries without any delay!
error <code> <pdu|-> <message|->
Issues an error report. The pdu contains all real fields comma separated.
notify
Send a SERIAL NOTIFY to all clients.
reset
Send a CACHE RESET to all clients.
quit, exit
Quits the loop and terminates the server
clients
Lists all clients
run <filename>
Executes a file line-by-line
sleep <seconds>
Pauses execution for the given number of seconds

Special commands:

* Toggle between command auto completion using TAB or file browsing using TAB
\q Quit
\h Help

Client Tools

Client tools are mainly used to test against a server. Not really of use for regular users, more a developer tool.

srxsvr_client – test SRx-Server API proxy

This tool is an example implementation of the SRx-Server-Client. It helps to test functions of the SRx server without the need of a full-blown API implementation such as QuaggaSRx. In addition, this test harness also provides a statistics framework that allows measurement of the performance of the SRx Server from its client's perspective.
To operate this test harness, a command line console is provided, with a set of commands to connect/disconnect to a BGP-SRx server, to send validation requests, etc. It implements most of the API and can be used as an example implementation for someone who wants to use the SRx-Server API.
The Help command provides a list of commands this client can send. Also, most commands with parameters can be used with default values. In example the connect command, entered with an incomplete set of attributes will request the necessary

parameters and provides default values. In addition, this tool allows code completion using the tabulator key.

Console commands:

Each command can be called without providing parameters. Missing parameters will be requested.

LOG_LEVEL <#>
: set the log level of this test harness.
(3)=ERROR, (4)=WANRING, (5)=NOTICE, (6)=INFO, (7)=DEBUG

RESET_PROXY <ip/#>
: Create a new proxy instance and set its default proxy ID.

NON_BLOCKING_SOCKET <true|false>
: As long as the proxy is not connected the socket type can be changed. Changing the socket type also changed the operational mode of the SRx-API itself. Only an external controlled proxy provides a non-blocking socket.

connect <host> <port> <proxy-id> <peer-as> [<peer-as>*] <0>
: Connect to the SRx-server using the provided information.

disconnect
: Disconnect from the SRx-server

reconnect
: Disconnects and re-connects.

addPeer <asn> [<asn>*] <0>
: Add the given peer or peers. The last peer MUST be 0.

delPeer <asn> [<asn>*] <0>
: Add the given peer or peers. The last peer MUST be 0.

verify <localID> <method> <asn> <prefix> <def-oval> <def-pval> <string>
: Send a verification request to the SRx server.
localID: an ID > 0
method: 0=just store, 1=ROA only, 2=BGPSec only, 3=both
asn: The origin AS
prefix: The prefix information
def-oval: The default origin validation result
0=valid, 1=unknown, 2=invalid, 3=undefined
def-pval: The default path validation result
0=valid, 2=invalid, 3=undefined
string: Some text string to simulate the remaining BGP update

byte string. This parameter might be changed in future versions.

sign <update-id> <prepend-counter> <peer-as>
Accepted but not yet implemented by SRx Server

delete <keepWindow> <update-id>
Request to delete the provided update after keepWindow seconds. The SRx-server will remove the update-client association but depending on memory need or other linking deletes the update at any time or not at all.

run <script-file>
Executes the commands found in the provided script. (Console output is suppressed during script runs – except statistic prints)

stat-init
Initialized the statistics framework

stat-mak <#>
Set a trigger when the statistics are generated. This command sets the trigger for a given number of received notifications.

stat-mark-nr <#>
Set a trigger when the statistics are generated. This command sets the trigger for a given number of received notifications except receipt notifications.

stat-print
Print the statistics. Will be done automatically if the stat-mark or stat-mark-nr trigger is set.

stat-start
Start the statistics framework.

stat-stop
Stop the statistics framework.

exit, quit, \q
Exit the program.

help
Help screen with quick explanation of the commands

rpkirtr_client – test rpki-rtr-protocol

Command: rpkirtr_client [<server-def:localhost> [<port-def:50001>]]
This tool allows testing a validation cache. In this case, it can be used as tester for the rpkirtr_svr tool. It helps to debug the RPKI cache test harness without the need of a full-blown BGP-SRx instance.

Support

Before contacting us, please verify that the SRx Server and its client are connected and properly communicating. We provide tools such as Wireshark plugins that allow analyzing the traffic in a human readable manner. Also, check firewall settings. If nothing helps (not even a reboot), please contact us and we will try to help. 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

Previous Developers:

Patrick Gleichmann (V0.1.0 only)