LINFO2142 - project

Release 2021

LINFO2142 students

Nov 11, 2021

CONTENTS:

CHAPTER

ONE

RPKI - FRROUTING

1.1 Introduction

Internet connections are nowadays more and more subject to attacks as its democratization increases. While in the early days of the Internet, communication protocols focused almost exclusively on the technical aspect of sending data, nowadays the security of transmissions must also be taken into account. While most of the protocols used today have received a security upgrade to protect them, such as SSL/TLS for HTTP, or have been replaced by other secure protocols, such as Telnet by SSH, other protocols are still not completely secure.

The BGP protocol used by border routers to communicate with external and autonomous network infrastructures is an example of a protocol that is still not completely secure everywhere. However, there are solutions that are increasingly implemented in the industry. In this tutorial, we will explain how to configure an autonomous system to secure its BGP connections with the outside world. In this way, the received IP prefixes can be authenticated. BGP hijacking is thus avoided.

1.1.1 What is RPKI ?

RPKI (*Resource Public Key Infrastructure*) is a specialized public key infrastructure (PKI) framework used to improve security of BGP messages.

RPKI is used to verify that a shared prefix comes from an identified AS. This verification is done using ROA certificates.

This infrastructure is composed of authorities that issue validation certificates, local cache servers that retrieve lists of certificates from multiple authorities, and clients that use these local servers to authenticate their incoming and outgoing BGP messages.

The local cache servers are themselves divided into two entities (this may vary depending on the implementation) a data server receiving the certificates, and another server communicating with this database server. They communicate with each other using a particular protocol, the RTR (*RPKI to Router*) protocol.

1.1.2 Why use RPKI ?

RPKI is used to improve the security of BGP messages by verifying the authenticity of advertised IP prefixes. The ASes through which the messages containing the IP prefixes passed and were retransmitted are authenticated via the public key infrastructures. This way, if an attacker tries to hijack BGP by sending wrong IP prefixes, the client will know from the Certificate Authority (CA) that these prefixes are not from the right AS.

IP prefix hijacking can be done in a number of ways, the two most common are:

- Regular prefix hijacking: occurs when the attacker's router hijacks a route from an existing IP prefix by advertising a new one. This prefix will then partially pollute the Internet, changing the routing tables based on the

preferability of the bogus route over the valid route. This will depend on the configuration of each particular network.

- Sub-prefix hijacking: occurs when the attacker steals a subnet of a prefix already existing in the routing tables, by advertising a route for the subnet from the attacker's network. Because of the plus-grand-prefix-matching policies of network operators, most networks on the Internet become polluted.

To further complicate matters, some stealth attackers may disguise both types of attacks with falsified AS paths without changing the original AS, while passing traffic through the attacker's network. [6] Certification of ASes and IP prefixes by Certification Authorities via RPKIs counters most attacks of this style.

For more information about RPKIs, you can consult the dedicated page *RPKI Architecture*.

1.2 RPKI Architecture

The deployment structure of RPKI is composed of 3 main elements.

1. Trust anchors
2. Local caches / Validator
3. Routers

1.2.1 Trust anchors

The trust anchors are a set of servers on which the authorative data of the RPKI are published. For a relying party to use RPKI, it has to choose a set of trust anchors from which it will retreive the RPKI data. According to RFC6480, an obvious choice would be the five RIRs (regional internet registries) which are AFRINIC, ARIN, APNIC, LACNIC and RIPE NCC [3].

The trust anchors are contacted via rsync[1] using information contained in a TAL. A TAL or trust anchor locator is composed of a rsync URI and a public key corresponding to a particular trust anchor as described in RFC6490. Here is an example:

rsync://rpki.example.org/rpki/hedgehog/root.cer
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAovWQL2lh6knDx
GUG5hbtCXvvh4AOzjhDkSHlj22gn/1oiM9IeDATIwP44vhQ6L/xvuk7W6
Kfa5ygmqQ+xOZOwTWPcrUbqaQyPNxokuivzyvqVZVDecOEqs78q58mSp9
nbtxmLRW7B67SJCBSzfa5XpVyXYEgYAjkk3fpmefU+AcxtxvvHB5OVPIa
BfPcs80ICMgHQX+fphvute9XLxjfJKJWkhZqZ0v7pZm2uhkcPx1PMGcrG
ee0WSDC3fr3erLueagpiLsFjwwpX6F+Ms8vqz45H+DKmYKvPSstZjCCq9
aJ0qANT9OtnfSDOS+aLRPjZryCNyvvBHxZXqj5YCGKtwIDAQAB

[1] https://rsync.samba.org/

1.2.2 Local caches / Validator

The role of the validator will firstly be to contact to choosen set of trust anchors in order to retrieve the RPKI data that will be needed to create a local ROA cache. And secondly, the validator will be used by the routers to access ROAs information. The validator has to periodically refresh the RPKI data.

1.2.3 Routers

RPKI enabled routers can contact the validator in order to verify that a route to a given IP prefix was originated by an AS that was allowed by the prefix holder. Depending on the validation result, the router can handle the received route accordingly. The communication bewteen a router and the validator is done via the RTR protocol.

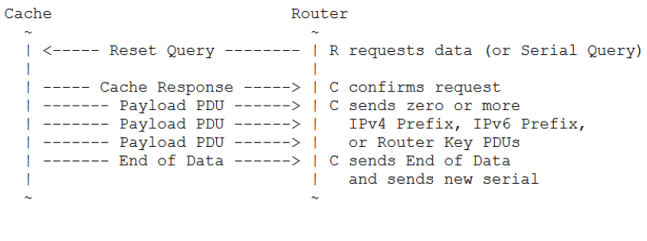
1.3 RTR protocol

The Resource Public Key Infrastructure (RPKI) to Router Protocol is a protocol used to deliver validated prefix origin data to the routers. It is defined in RFC6820. RTR was designed to be a lightweight protocol with a low memory footprint. The core of the protocol is to synchronize a database between a validator and a router. It is based on incremental transfers and runs on a TCP connection. Below is a quick description of the important protocol data units that can be exchanged bewteen a cache and a router.

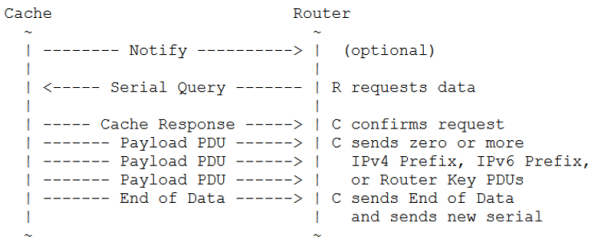
- **Serial Notify**: the cache notifies the routers that the cache has new data.
- **Serial Query**: the router asks for all new announcements and withdrawals.
- **Reset Query**: the router wants to receive the complete content of the database.
- **Cache response**: the cache will answer to a serial query or a reset query with a cache response. This respone is followed by zero or more PDUs payloads and an End of data PDU.
- **IPv4 prefix**: this payload PDU announces an IPv4 prefix - AS number pair.
- **IPv6 prefix**: this payload PDU announces an IPv6 prefix - AS number pair.
- **End of Data**: the cache will send this PDU to announce that the cache response is over.
- **Cache reset**: the cache can answer this PDU to a serial query to inform the router that it is unable to provide an incremental update.
- **Error report**: the cache or the router uses this PDU to report an error to the other.

The typical sequence of PDU exchanges falls in the four diagrams shown below:

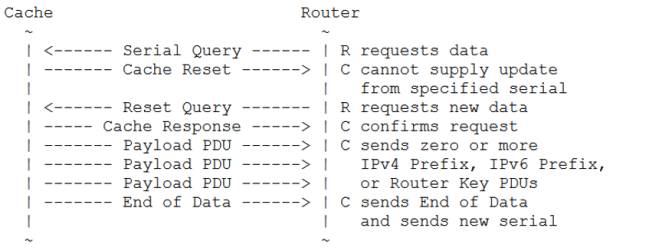
Start or Restart



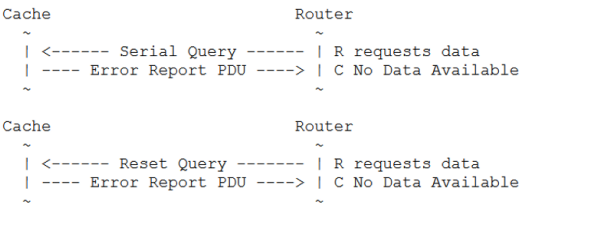
Typical Exchange



No Incremental Update Available



Cache Has No Data Available



1.4 Route Origin Authorizations

A Route Origin Authorization (ROA) is a digitally signed object that allows to verify that an IP address block holder has allowed an Autonomous System to annoucne one or more prefix falling in the address block. More precisely, the content of a ROA identifies one AS that has been authorized by the addres block holder to advertise route to IP prefix from this address block. If the block holder wants to authorize multiple ASs to advertise the same prefix, it has to issue multiple ROAs [4].

A ROA follows the following structure:

> **RouteOriginAttestation ::= SEQUENCE {** version [0] INTEGER DEFAULT 0, asID ASID, ipAddrBlocks SEQUENCE (SIZE(1..MAX)) OF ROAIPAddressFamily }

It is composed of 3 fields:

1. version: the version number, must be 0.
2. asID: an integer representing the AS number receiving the Authorization.
3. ipAddrBlocks: the set of IP prefixes that the AS is allowed to announce.

1.5 Validation of Route Origination

As mentionned before, the interest of RPKI is to ensure that a route to a prefix was annouced over BGP by an AS that was allowed to announce such prefix. Therefore, a validation process has to take place for each route received over a BGP session in order to check the IP prefix - AS number pair. It is performed using the ROA information and can have 3 different outcomes: valid, unknown or invalid. The validation process is the following:

1. Firstly, a set of ROAs matching the IP prefix or being a covering aggregate of the IP prefix of the route is selected. This set forms “the candidate ROAs”.
2. If the set of candidate ROAs is empty, the validation procedure stops and the outcome is “unknown”.
3. If any of the candidate ROA contains an asID field matching the route’s origin AS and the route IP prefix matches an IP prefix in the ROA, the outcome of the procedure is “valid”.

4. If this step is reached, the outcome of the procedure is “invalid”.

The table below summurizes the procedure:

Route AS-> Prefix	matching AS	**non-matching** AS
Non- Intersecting	unknown	unknown
Covering Aggregate	unknown	unknown
match ROA prefix	valid	invalid
More Specific than ROA	invalid	invalid

Once the validation is done, it can be applyed to the route selection process. The preference order should be the following: “valid” is to be preferred over “unknown”, which is to be preferred over “invalid”. However, the action to be taken for an “unknown” or “invalid” route’s validity state is a matter of local routing policy. As mentionned in RFC6483 [2], it is recommended to not consider “unknown” validity state as sufficient ground to reject a route from futher consideration due to the partial use of ROAs.

1.6 Network configuration

1.6.1 FRRouting

FRRouting is a free and open-source Internet routing protocol suite for Linux and Unix platforms, it implements many protocoles such as BGP or OSPF.

We will assume that an implementation using FRRouting is already installed on your machine.

The BGP commands to use RPKI are already implemented in FRRouting. They are available in full on the official documentation site[1] . The implementation of RPKI by FRRouting is based on the definition given in RFC 6810 and the validation scheme on RFC 6811 [1, 5].

You need to install the additional package “frr-rpki-rtrlib” to enable support for RPKI, otherwise the “bgpd” daemon will simply not start.

Configure a router

To activate the RPKI configuration mode on a router on which you have previously logged in, simply activate the “rpki” command. Be careful, activating this command does not activate the validation of IP prefixes in itself. In order to work, at least one cache server must be available (see RPKI validator below).

1.6.2 RPKI validator

An essential element required to use RPKI is a local cache / validator. There are multiple open source implementations available[2] . We decided to go with OctoRPKI made by Cloudflare. It is composed of two elements: OctoRPKI and GoRTR.

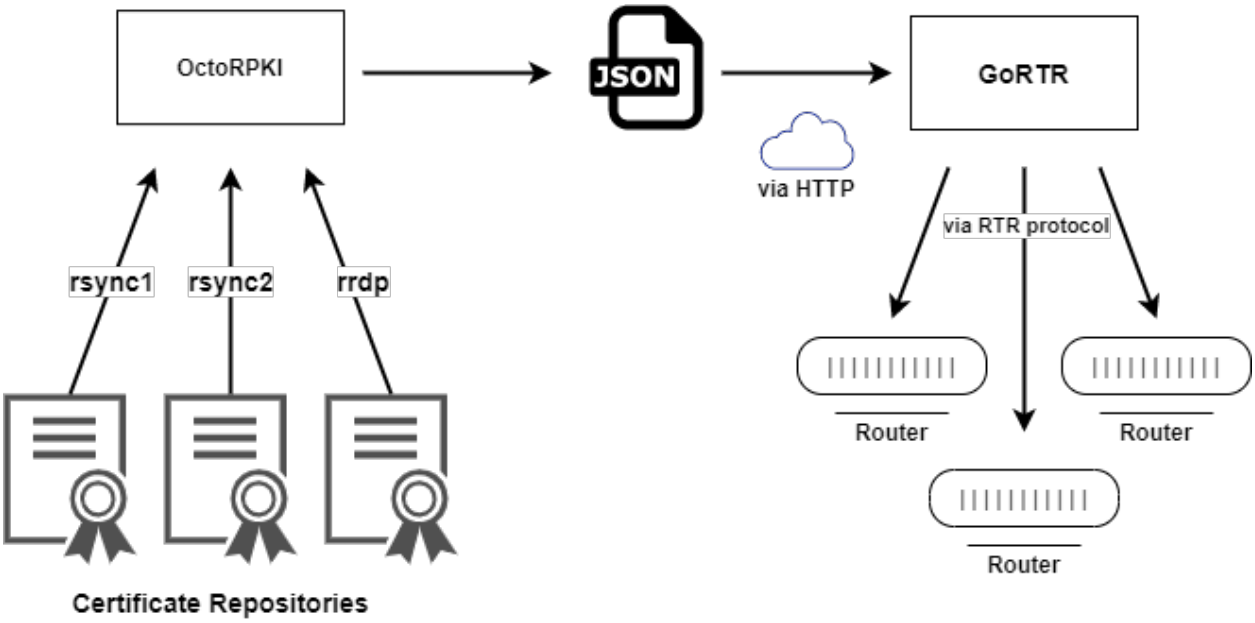
[1] http://docs.frrouting.org/en/latest/bgp.html#prefix-origin-validation-using-rpki
[2] https://labs.ripe.net/author/tashi_phuntsho_3/how-to-install-an-rpki-validator/

OctoRPKI

The role of OctoRPKI is to retrieve the RPKI data from the trust anchors and output a JSON file containing all the IP prefix - AS number pairs. It will automatically update the JSON file depending on a given refresh period.

GoRTR

GoRTR is an implementation of a RTR server and will assure the communication between the routers and the cache. It contacts the OctoRPKI instance in order to retrieve the JSON file.



CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] R. Bush and R. Austein. The resource public key infrastructure (rpki) to router protocol. RFC 6810, RFC Editor, January 2013.

[2] G. Huston and G. Michaelson. Validation of route origination using the resource certificate public key infrastructure (pki) and route origin authorizations (roas). RFC 6483, RFC Editor, February 2012.

[3] G. Huston, S. Weiler, G. Michaelson, and S. Kent. Resource public key infrastructure (rpki) trust anchor locator. RFC 6490, RFC Editor, February 2012.

[4] M. Lepinski, S. Kent, and D. Kong. A profile for route origin authorizations (roas). RFC 6482, RFC Editor, February 2012.

[5] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. Bgp prefix origin validation. RFC 6811, RFC Editor, January 2013. http://www.rfc-editor.org/rfc/rfc6811.txt. URL: http://www.rfc-editor.org/rfc/rfc6811.txt.

[6] Zheng Zhang, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. Practical defenses against bgp prefix hijacking. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07. New York, NY, USA, 2007. Association for Computing Machinery. URL: https://doi.org/10.1145/1364654.1364658, doi:10.1145/1364654.1364658.