|  |
| --- |
| **RPKI** |

|  |
| --- |
| There are two types of Network Admins who deal with BGP, those who have created an |

|  |
| --- |
| international incident and/or outage, and those who are lying |

|  |
| --- |
| —tweet by EvilMog, 2020-02-21 |

RPKI is a framework PKI designed to secure the Internet routing infrastructure. It

associates BGP route announcements with the correct originating ASN which BGP

routers can then use to check each route against the corresponding ROA for validity.

|  |
| --- |
| RPKI is described in **RFC 6480**. |

|  |
| --- |
| A BGP-speaking router like VyOS can retrieve ROA information from RPKI “Relying |

Party software” (often just called an “RPKI server” or “RPKI validator”) by

using RTR protocol. There are several open source implementations to choose from,

such as NLNetLabs’ Routinator (written in Rust),

Cloudflare’s GoRTR and OctoRPKI (written in Go), and RIPE NCC’s

RPKI Validator (written in Java). The RTR protocol is described in **RFC 8210**.

|  |
| --- |
| **Tip** |

|  |
| --- |
| If you are new to these routing security technologies then there is an excellent guide to |

|  |
| --- |
| RPKI by NLnet Labs which will get you up to speed very quickly. Their documentation |

|  |
| --- |
| explains everything from what RPKI is to deploying it in production. It also has some help |

|  |
| --- |
| and operational guidance including “What can I do about my route having an Invalid |

|  |
| --- |
| state?” |
| **Getting started** |

|  |
| --- |
| First you will need to deploy an RPKI validator for your routers to use. The RIPE NCC |

|  |
| --- |
| helpfully provide some instructions to get you started with several different options. |

|  |
| --- |
| Once your server is running you can start validating announcements. |

|  |
| --- |
| Imported prefixes during the validation may have values: |

|  |
| --- |
| **valid** |

The prefix and ASN that originated it match a signed ROA. These are probably

|  |
| --- |
| trustworthy route announcements. |

|  |
| --- |
| **invalid** |

|  |
| --- |
| The prefix or prefix length and ASN that originated it doesn’t match any |

existing ROA. This could be the result of a prefix hijack, or merely a

misconfiguration, but should probably be treated as untrustworthy route

|  |
| --- |
| announcements. |

|  |
| --- |
| **notfound** |

|  |
| --- |
| No ROA exists which covers that prefix. Unfortunately this is the case for |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Note** | | about 80% of the IPv4 prefixes which were announced to the DFZ at the start of 2020 |

|  |
| --- |
| If you are responsible for the global addresses assigned to your network, please make |

|  |
| --- |
| sure that your prefixes have ROAs associated with them to avoid being *notfound* by |

|  |
| --- |
| RPKI. For most ASNs this will involve publishing ROAs via your RIR (RIPE NCC, APNIC, |

|  |
| --- |
| ARIN, LACNIC or AFRINIC), and is something you are encouraged to do whenever you |

|  |
| --- |
| plan to announce addresses into the DFZ. |

|  |
| --- |
| Particularly large networks may wish to run their own RPKI certificate authority and |

|  |
| --- |
| publication server instead of publishing ROAs via their RIR. This is a subject far beyond |

|  |
| --- |
| the scope of VyOS’ documentation. Consider reading about Krill if this is a rabbit hole |

|  |
| --- |
| you need or especially want to dive down. |
| **Features of the Current Implementation** |

|  |  |
| --- | --- |
| In a nutshell, the current implementation provides the following features: | |
|  | The BGP router can connect to one or more RPKI cache servers to receive |

|  |
| --- |
| validated prefix to origin AS mappings. Advanced failover can be implemented |

|  |  |
| --- | --- |
|  | by server sockets with different preference values.  If no connection to an RPKI cache server can be established after a pre- |

|  |
| --- |
| defined timeout, the router will process routes without prefix origin validation. |

|  |
| --- |
| It still will try to establish a connection to an RPKI cache server in the |

background.

|  |  |
| --- | --- |
|  | By default, enabling RPKI does not change best path selection. In particular, |

|  |
| --- |
| invalid prefixes will still be considered during best path selection. However, |

|  |  |
| --- | --- |
|  | the router can be configured to ignore all invalid prefixes.  Route maps can be configured to match a specific RPKI validation state. This |

|  |
| --- |
| allows the creation of local policies, which handle BGP routes based on the |

|  |  |
| --- | --- |
|  | outcome of the Prefix Origin Validation.  Updates from the RPKI cache servers are directly applied and path selection is |

|  |
| --- |
| updated accordingly. (Soft reconfiguration must be enabled for this to work). **Configuration** |

|  |
| --- |
| **set protocols rpki polling-period <1-86400>** |

|  |
| --- |
| Define the time interval to update the local cache |

|  |
| --- |
| The default value is 300 seconds. |

|  |
| --- |
| **set protocols rpki cache <address> port <port>** |

|  |
| --- |
| Defined the IPv4, IPv6 or FQDN and port number of the caching RPKI caching instance |

|  |
| --- |
| which is used. |

|  |
| --- |
| This is a mandatory setting. |

|  |
| --- |
| **set protocols rpki cache <address> preference <preference>** |

|  |
| --- |
| Multiple RPKI caching instances can be supplied and they need a preference in which |

|  |
| --- |
| their result sets are used. |

|  |
| --- |
| This is a mandatory setting. |

|  |
| --- |
| **SSH** |

|  |
| --- |
| Connections to the RPKI caching server can not only be established by HTTP/TLS but |

|  |  |
| --- | --- |
| you can also rely on a secure SSH session to the server. To enable SSH, first you need to create an SSH client keypair using generate ssh client-key /config/auth/id\_rsa\_rpki . Once your   |  | | --- | |  | |

|  |
| --- |
| key is created you can setup the connection. |

|  |
| --- |
| **set protocols rpki cache <address> ssh username <user>** |

|  |
| --- |
| SSH username to establish an SSH connection to the cache server. |

|  |
| --- |
| **set protocols rpki cache <address> ssh known-hosts-file <filepath>** |

|  |
| --- |
| Local path that includes the known hosts file. |

|  |
| --- |
| **set protocols rpki cache <address> ssh private-key-file <filepath>** |

|  |
| --- |
| Local path that includes the private key file of the router. |

|  |
| --- |
| **set protocols rpki cache <address> ssh public-key-file <filepath>** |

|  |
| --- |
| Local path that includes the public key file of the router. |

|  |
| --- |
| **Note** |

|  |
| --- |
| When using SSH, known-hosts-file, private-key-file and public-key-file are mandatory |

|  |
| --- |
| options. |
| **Example** |

|  |
| --- |
| We can build route-maps for import based on these states. Here is a simple RPKI |

|  |
| --- |
| configuration, where *routinator* is the RPKI-validating “cache” server with ip *192.0.2.1*: |

|  |
| --- |
| set protocols rpki cache 192.0.2.1 port '3323'  set protocols rpki cache 192.0.2.1 preference '1' |

Here is an example route-map to apply to routes learned at import. In this filter we reject

prefixes with the state *invalid*, and set a higher *local-preference* if the prefix is

RPKI *valid* rather than merely *notfound*.

|  |
| --- |
| set policy route-map ROUTES-IN rule 10 action 'permit'  set policy route-map ROUTES-IN rule 10 match rpki 'valid'  set policy route-map ROUTES-IN rule 10 set local-preference '300' set policy route-map ROUTES-IN rule 20 action 'permit'  set policy route-map ROUTES-IN rule 20 match rpki 'notfound' set policy route-map ROUTES-IN rule 20 set local-preference '125' set policy route-map ROUTES-IN rule 30 action 'deny'  set policy route-map ROUTES-IN rule 30 match rpki 'invalid' |

|  |
| --- |
| Once your routers are configured to reject RPKI-invalid prefixes, you can test whether |

|  |
| --- |
| the configuration is working correctly using the RIPE Labs RPKI Test experimental tool. |

Previous Next