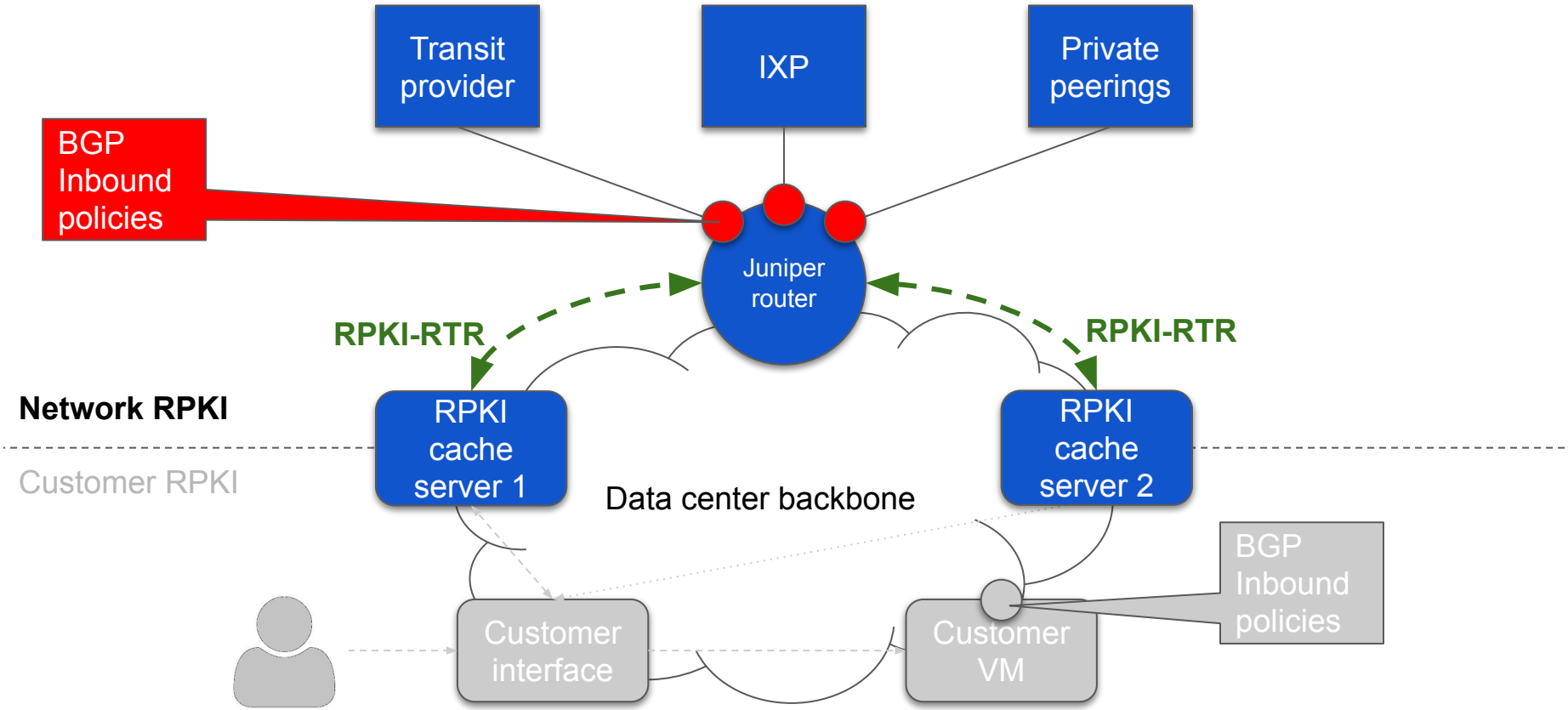
Configuring RPKI in Juniper routers

Tomas Lynch
Senior Network Architect
Vultr

Basic Topology



RPKI cache server

RPKI cache server

NLnet Labs Routinator 3000

- Connects to the Trust Anchors of the five RIRs
- Downloads all of the certificates and ROAs and validates the signatures
- Feeds the validated information to routers

Let’s call them validators

Network RPKI

Juniper configuration

Two main configurations

1) Sessions against RPKI validators

2) Policies against peer BGP sessions

Juniper Origin Validation for BGP

```
routing-options {
    validation {
        group <group name> {
            session <server1 IP> {
            }
            session <server2 IP> {
                port <port number>; # optional
                local-address <source address>; # optional
            }
        }
    }
}
```

Origin Validation for BGP example

```
routing-options {
    validation {
        group VALIDATOR {
            session 2001:db8::1;
            session 2001:db8::2;
        }
    }
}
```

- Default port 2222
- Sessions are “dual stack”: information for v4 and v6 is received independent of the server IP address version

Verifying validator session

```
router> show validation session

Session         State  Flaps Uptime          #IPv4/IPv6 records

2001:db8::1    Up         0  5w4d 07:55:29 137130/23053

2001:db8::2    Up         0 10w1d 00:49:40 137130/23053
```

Route validation records

On the router, the database entries are formatted as route validation (RV) records

An RV record is a (prefix, maximum length, origin AS) triple. One per validator.

```
router> show validation database
RV database for instance master
Prefix                 Origin-AS Session       State   Mismatch
...
2001:260::/32-48            2518 2001:db8::1   valid
2001:260::/32-48            2518 2001:db8::2   valid
2001:288::/32-32            1659 2001:db8::1   valid
2001:288::/32-32            1659 2001:db8::2   valid
2001:2f0::/32-128           7514 2001:db8::1   valid
2001:2f0::/32-128           7514 2001:db8::2   valid
...
```

Route validation mismatch

```
router> show validation database mismatch
RV database for instance master
Prefix                  Origin-AS Session     State   Mismatch
<snip>
2001:648:25e0::/48-48       47616 2001:db8::1 valid   *

2001:648:25e0::/48-48       47616 2001:db8::2 valid   *

2001:648:25e0::/48-48      203348 2001:db8::1 valid   *

2001:648:25e0::/48-48      203348 2001:db8::2 valid   *
```

The `mismatch` qualifier is useful for finding conflicting origin-autonomous-system information between RPKI caches.

Show RV records by Autonomous System

```
router> show validation database origin-autonomous-system 6140
RV database for instance master

Prefix                  Origin-AS Session     State   Mismatch
107.161.208.0/20-20          6140 2001:db8::1 valid
107.161.208.0/20-20          6140 2001:db8::2 valid
200.14.34.0/24-24            6140 2001:db8::1 valid   *
200.14.34.0/24-24            6140 2001:db8::2 valid   *
2001:470:111::/48-48         6140 2001:db8::1 valid
2001:470:111::/48-48         6140 2001:db8::2 valid
2602:ffd3::/36-36            6140 2001:db8::1 valid
2602:ffd3::/36-36            6140 2001:db8::2 valid

  IPv4 records: 4
  IPv6 records: 4
```

Show RV records statistics

```
router> show validation statistics
Total RV records: 321114
Total Replication RV records: 321114
  Prefix entries: 147081
  Origin-AS entries: 160560
Memory utilization: 51298803 bytes
Policy origin-validation requests: 175890241
  Valid: 51026232
  Invalid: 395761
  Unknown: 124468248
BGP import policy reevaluation notifications: 708595
  inet.0, 620046
  inet6.0, 88549
```

Validating routes

Validation against peers is done using BGP policies

Validation has three states:

Valid: the prefix has an RV and matches the autonomous system

Invalid: the prefix has an RV but does not match the autonomous system or the prefix length is longer than the maximum accepted

Unknown: the prefix does not have an RV entry

A fourth state called **Unverified** is used when the prefix didn’t run on a validation policy

Basic policy configuration [1/3]

```
[edit policy-options policy-statement PEER_INBOUND_POLICY]
term RPKI_INVALID {
    from {
        protocol bgp;
        validation-database invalid;
    }
    then {
        validation-state invalid;
        community set origin-validation-state-invalid;
        reject;
    }
}
...
```

Basic policy configuration [2/3]

```
...
term RPKI_UNKNOWN {
    from {
        protocol bgp;
        validation-database unknown;
    }
    then {
        validation-state unknown;
        community set origin-validation-state-unknown;
        then accept;
    }
}
...
```

Basic policy configuration [3/3]

```
...
term RPKI_VALID {
    from {
        protocol bgp;
        validation-database valid;
    }
    then {
        validation-state valid;
        community set origin-validation-state-valid;
        then accept;
    }
}
```

Supporting configuration (optional)

```
set policy-options community origin-validation-state-invalid
members 0x4300:0.0.0.0:2

set policy-options community origin-validation-state-unknown
members 0x4300:0.0.0.0:1

set policy-options community origin-validation-state-valid
members 0x4300:0.0.0.0:0
```

Applying BGP policy to peer

```
set protocols bgp group MY_PEER import PEER_INBOUND_POLICY
```

Route Validation state

```
router> show route validation-state ?
Possible completions:
  invalid              Invalid route validation state
  unknown              Unknown route validation state
  unverified           Unverified route validation state
  valid                Valid route validation state
```

Route validation state examples

```
router> show route validation-state valid

2001:db8::/32      *[BGP/170] 3d 03:43:17, MED 0, localpref 70
                      AS path: 64496 I, validation-state: valid
                    > to 2001:db8::10 via et-0/0/0

router> show route validation-state invalid hidden

2001:db8::/32       [BGP/170] 3d 03:43:17, MED 0, localpref 70
                      AS path: 64496 I, validation-state: invalid
                      Discard
```

Route validation state examples

```
router> show route validation-state unknown

2001:db8::/32      *[BGP/170] 3d 03:43:17, MED 0, localpref 70
                      AS path: 64496 I, validation-state: unknown
                    > to 2001:db8::10 via et-0/0/0

router> show route validation-state unverified

2001:db8::/32      *[BGP/170] 3d 03:43:17, MED 0, localpref 70
                      AS path: 64496 I, validation-state: unverified
                    > to 2001:db8::10 via et-0/0/0
```

Route validation state (7/7/2020)

```
router> show route validation-state invalid hidden | grep
"^[1-9].*" | count
Count: 3811 lines (0.43%)

router> show route validation-state valid | grep "^[1-9].*"
| count
Count: 192587 lines (21.66%)

router> show route validation-state unknown | grep
"^[1-9].*" | count
Count: 692782 lines (77.91%)
```

Route validation per IP version

<table>

Variations on RPKI_INVALID term

```
[edit policy-options policy-statement PEER_INBOUND_POLICY]
term RPKI_INVALID {
    from {
        protocol bgp;
        validation-database invalid;
    }
    then {
        validation-state invalid;
        community set origin-validation-state-invalid;
        next-hop discard;
        accept;
    }
}
...
```

Variations on RPKI_INVALID term (cont.)

```
router> show route 2001:db8::/32

2001:db8::/32          [BGP/170] 3d 03:43:17, MED 0, localpref 70
                         AS path: 64496 I, validation-state: invalid
                         Discard
```

- Installed on the routing table
- Useful when receiving default route from transit providers

What happens if no validators?

It works when validators are down

Add a final term to the policy statement to accept prefixes based on different rules

```
term RPKI_INVALID {
  from validation-database invalid;
  then reject;
}
term RPKI_UNKNOWN {
  from validation-database unknown;
  then accept;
}
term RPKI_VALID {
  from validation-database valid;
  then accept;
}
term DEFAULT_ACTION {
  from <insert your rules here>;
  then accept;
```

Juniper validation considerations

- Junos OS Release 12.2
- RPKI validation is available only in the master instance.
  - If configured in a routing instance, then error message **`RV instance is not running`**.
- When you enable the RPKI authentication
  - Junos OS opens the TCP port 2222 automatically without any notice.
  - You can apply a filter to block and secure this port.
- Supported in a few platforms only, check Origin validation for BGP
  - However I have it running on a QFX

References

- Routinator 3000
- Configuring Origin Validation for BGP
- RFC6810 - RPKI RTR
- show validation database
- BGP Origin Validation Using Resource Public Key Infrastructure
- Origin validation for BGP

Thank you!