Acknowledgements

- This material includes valuable contributions by Randy Bush, Mark Tinka, Aftab Siddiqui, Tashi Phuntsho, Warrick Mitchell and others.

- Use of these materials is encouraged as long as the source is fully acknowledged and this notice remains in place.

- Bug fixes and improvements are welcomed.
  - Please email workshop (at) bgp4all.com

Philip Smith
BGP Videos

- NSRC has produced a library of BGP presentations (including this one), recorded on video, for the whole community to use
  - https://learn.nsrc.org/bgp
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
Why Origin Validation?

The trust model of BGP
Validating BGP Route Announcements

- How do we know that an AS is permitted to originate the prefix it is originating?
- Implicit trust?
- Because the Internet Routing Registry says so?
  - The Internet Routing Registry (IRR) only documents routing policy
  - And has a large amount of outdated/incorrect information
- Is there something else?
  - Yes: Route Origin Authorisation
BGP – Why Origin Validation?

- Prevent YouTube accident & Far Worse
  - Almost every day there is an incident of prefix hijack somewhere on the Internet

- Prevents most accidental announcements
  - “Fat finger”, missing BGP policy configuration, etc

- Does not prevent malicious path attacks
  - Example: alteration of AS-PATH attribute along the announcement chain
  - That requires ‘Path Validation’, using BGPsec
RPKI

- RPKI – Resource Public Key Infrastructure
  - The Certificate Infrastructure for origin and path validation

- We need to be able to authoritatively prove who owns an IP prefix and which AS(s) may announce it
  - Prefix ownership follows the allocation hierarchy
  - IANA → RIRs → ISPs → etc
What is RPKI?

- **Resource Public Key Infrastructure (RPKI)**
  - A security framework for verifying the association between resource holder and their Internet resources
  - Created to address the issues discussed in RFC 4593 “Generic Threats to Routing Protocols” (Oct 2006)

- **Helps to secure Internet routing by validating routes**
  - Proof that prefix announcements are coming from the legitimate holder of the resource
  - RFC 6480 – An Infrastructure to Support Secure Internet Routing (Feb 2012)
  - RFC 7115 – Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)
Benefits of RPKI for Routing

- Prevents route hijacking
  - A prefix originated by an AS without authorisation
  - Reason: malicious intent

- Prevents mis-origination
  - A prefix that is mistakenly originated by an AS which does not own it
  - Also, route leakage
  - Reason: configuration mistake / fat finger
BGP Security (BGPsec)

- Extension to BGP that provides improved security for BGP routing
  - Published as RFC8205
  - Not yet deployed
- Implemented via a new optional non-transitive BGP attribute (BGPsec_PATH) that contains a digital signature
- BGPsec supplements BGP origin validation
  - Allows routers to generate, propagate, and validate BGP update messages with the BGPsec_PATH attribute set
BGPsec Components

- **Origin Validation**
  - Using the RPKI to detect and prevent mis-originations of someone else’s prefixes (RFC6483)
  - Implementation started in 2012

- **AS-Path Validation**
  - BGPsec has not yet begun deployment (cryptographic computation load)
  - soBGP was one early option
    - [https://datatracker.ietf.org/doc/draft-white-sobgp-architecture/](https://datatracker.ietf.org/doc/draft-white-sobgp-architecture/) (expired)
    - Not standardised or implemented
  - ASPA (Autonomous System Provider Authorisation) is one more step towards full BGPsec deployment
RPKI Nomenclature

- **Issuing Party**
  - The entity operating as certificate authority (CA)

- **Trust Anchor**
  - The authority from which trust is assumed, rather than derived from intermediates – the root of the tree

- **Relying Party**
  - The operator system gathering data from the certificate authority to be used for validation

- **Route Origin Authorisation**
  - A digital object linking an AS number with the IP address space it is authorised to originate
Issuing Party

- Internet Registries (RIR, NIR, Large LIRs)
- Acts as a Certificate Authority and issues certificates for customers
- Provides a web interface to issue ROAs for customer prefixes
- Publishes the ROA records
Relying Party (RP)

RP Cache software also known as a Validator
Each of the RIRs publishes their “Trust Anchor Locator” (TAL) – the file that contains both the URL of the RPKI repository and the public key.
RPKI Service Models

- **Hosted Model:**
  - The RIR runs the CA on behalf of its members
    - Manage keys, repository, etc
    - Generate certificates for resource certifications

- **Delegated Model:**
  - Member becomes the CA, delegated from the parent CA (the RIR)
    - Operates the full RPKI system
    - Several entities now operating delegated CAs
  - **CA Software**
    - NLnetLabs Krill: https://www.nlnetlabs.nl/projects/rpki/krill/
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
Route Origin Authorisation
Route Origin Authorisation (ROA)

- A digital object that contains a list of address prefixes and one AS number
- It is an authority created by a prefix holder to authorise an AS Number to originate one or more specific route advertisements
- Publish a ROA using your RIR member portal
  - Consult your RIR for how to use their member portal to publish your ROAs
Route Origin Authorisation

- A typical ROA would look like this:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>10.10.0.0/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Length</td>
<td>/18</td>
</tr>
<tr>
<td>Origin-AS</td>
<td>AS65534</td>
</tr>
</tbody>
</table>

- There can be more than one ROA per address block
  - Allows the operator to originate prefixes from more than one AS
  - Caters for changes in routing policy or prefix origin
Creating ROAs

- Only create ROAs for the aggregate and the exact subnets expected in the routing table

Examples:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Max Length</th>
<th>Origin AS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.0.0/16</td>
<td>/24</td>
<td>65534</td>
<td>ROA covers /16 through to /24 – any announced subnets to /24 will be Valid if from AS65534</td>
</tr>
<tr>
<td>10.10.0.0/16</td>
<td>/16</td>
<td>65534</td>
<td>ROA covers only /16 – any announced subnets will be Invalid</td>
</tr>
<tr>
<td>10.10.4.0/22</td>
<td>/24</td>
<td>65534</td>
<td>ROA covers this /22 through to /24</td>
</tr>
<tr>
<td>10.10.32.0/22</td>
<td>/24</td>
<td>64512</td>
<td>Valid ROA covers /22 through to /24 announcements from AS64512</td>
</tr>
</tbody>
</table>
Creating ROAs – Important Notes

- Always create ROAs for the aggregate and the individual subnets being routed in BGP
- Example:
  - If creating a ROA for 10.10.0.0/16 and “max prefix” length is set to /16
    - There will only be a valid ROA for 10.10.0.0/16
    - If a subnet of 10.10.0.0/16 is originated, it will be state Invalid
Creating ROAs – BCP185

- RFC9319/BCP185
  - Avoid using maxLength attribute unless in special cases
  - Do NOT create ROAs for subnets of an aggregate unless they are actively routed
    - If ROA exists, but subnet is not routed, it leaves an opportunity for someone else to mis-originate the subnet using the valid origin AS, resulting in a hijack
  - Recommendation: Use minimal ROAs wherever possible – only for prefixes that are actually being announced
Creating ROAs – Important Notes

- Some current examples of problematic ROAs:

<table>
<thead>
<tr>
<th>AS</th>
<th>Prefix</th>
<th>Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>328037</td>
<td>2c0f:f0c8::/32</td>
<td>128</td>
</tr>
</tbody>
</table>

- This means that any and every subnet of 2C0F:F0C8::/32 originated by AS328037 is valid
  - An attacker can use AS328037 as their origin AS to originate 2C0F:F0C8:A0::/48 to deny service to that address block
  - Known as a validated hijack!

<table>
<thead>
<tr>
<th>AS</th>
<th>Prefix</th>
<th>Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>3462</td>
<td>1.34.0.0/15</td>
<td>24</td>
</tr>
</tbody>
</table>

- This means that any subnet of 1.34.0.0/15 down to a /24 as originated by AS3462 is valid
  - An attacker can use AS3462 as their origin AS to originate 1.34.10.0/24 to deny service to that address block
Creating ROAs: “Validated Hijack”

- If the 1.34.10.0/24 prefix had had no ROA, route origin validation would have dropped the invalid announcement at the upstream AS.

Diagram:
- Global Internet
- Viewer
- Upstream
- AS3462
- Originator of 1.34.0.0/15 with ROA MaxLen of /24
- Traffic Flow for 1.34.10.0/24
- Attacker: uses target AS as their origin
- Originates: 1.34.10.0/24

Valid ROA for /15 and /24
Best path selection: /24 preferred over the /15
Creating ROAs: pre-RIR Address Space

- Some entities were assigned address space by InterNIC
  - This is prior to the existence of the RIRs
- How to sign ROAs for these resources?
- Some RIRs will support the signing of legacy address space ROAs
  - If there is documentation proving the holding
  - If there is some service agreement for resources allocated by the RIR
  - Or by some other arrangement
  - Example, APNIC:
  - Example, RIPE NCC:
  - Example, ARIN:
    - LRSA (Legacy Registration Services Agreement) now permits signing of ROAs for legacy address space
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
Route Origin Validation
Route Origin Validation

- Router must support RPKI
- Checks an RP cache / validator
  - Uses RtR protocol, described in RFC8210
- Validation returns 3 states:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>When authorisation is found for prefix X coming from ASN Y</td>
</tr>
<tr>
<td>Invalid</td>
<td>When authorisation is found for prefix X but not from ASN Y, or not allowable subnet size</td>
</tr>
<tr>
<td>Not Found</td>
<td>When no authorisation data is found for prefix X</td>
</tr>
</tbody>
</table>
Route Origin Validation – AS0

- RFC6483 also describes “Disavowal of Routing Origination”
  - AS 0 has been reserved for network operators and other entities to identify non-routed networks
  - Which means:
    - “A ROA with a subject of AS0 (AS0 ROA) is an attestation by the holder of a prefix that the prefix described in the ROA, and any more specific prefix, should not be used in a routing context”

- Any prefixes with ROA indicating AS0 as the origin AS need to be dropped
  - If these prefixes appear with any other origin, their ROAs will be invalid, achieving this goal
Route Origin Validation – AS0

Possible use cases of AS0:

- Internal use of a prefix that should not appear in the global BGP table
- Internet Exchange Point LAN must never appear in the global BGP table
- Private Address space (IPv4) and non-Global Unicast space (IPv6)
- Unassigned address space
  - This is under discussion within the various RIR policy fora
- IPv4 and IPv6 address resources which should not appear in the global BGP table
  - For example, the special use address space described in RFC6890
Route Origin Validation – AS0

- APNIC & LACNIC have now published their AS0 TALs
  - Operated separately from the regular TAL
    - [https://www.apnic.net/community/security/resource-certification/trust-anchor-locator/](https://www.apnic.net/community/security/resource-certification/trust-anchor-locator/)
    - [https://www.lacnic.net/4984/2/lacnic/rpki-rpki-trust-anchor](https://www.lacnic.net/4984/2/lacnic/rpki-rpki-trust-anchor)
  - Simply add to the TAL folder in the validator cache

- Some examples of AS0 being used today:

<table>
<thead>
<tr>
<th>RPKI/RTR prefix table</th>
<th>Prefix</th>
<th>Prefix Length</th>
<th>Origin-AS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.57.180.0</td>
<td>22 - 24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5.57.80.0</td>
<td>22 - 22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>23.4.85.0</td>
<td>24 - 24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>23.173.176.0</td>
<td>24 - 24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>23.211.114.0</td>
<td>23 - 24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>45.12.44.0</td>
<td>22 - 22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>58.181.75.0</td>
<td>24 - 24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>109.122.244.0</td>
<td>22 - 22</td>
<td>0</td>
</tr>
</tbody>
</table>
Route Origin Validation – Implementations

- Cisco IOS – available from release 15.2
- Cisco IOS/XR – available from release 4.3.2
- Juniper JunOS – available from release 12.2
- Nokia – available from release R12.0R4
- Huawei – available from release V800R009C10
- FRR – available from release 4.0
- BIRD – available from release 1.6
- OpenBGPD – available from OpenBSD release 6.4
- GoBGP – available since 2018
- VyOS – available from release 1.2.0-RC11
- Mikrotik ROS – available from release v7
- Arista EOS – available from release 4.24.0F
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
Validator Cache

Choosing, deploying, and operating a Validator Cache
RPKI Validator Caches (1)

- **NLnet Labs Routinator 3000**
  - [https://www.nlnetlabs.nl/projects/rpki/routinator/](https://www.nlnetlabs.nl/projects/rpki/routinator/)
  - [https://github.com/NLnetLabs/routinator](https://github.com/NLnetLabs/routinator)
  - Packages available for Debian/Ubuntu, RHEL/CentOS & FreeBSD
  - (Can also be built from source)

- **LACNIC/NIC Mexico validator (FORT)**
  - [https://fortproject.net/en/validator](https://fortproject.net/en/validator)
  - [https://nicmx.github.io/FORT-validator/](https://nicmx.github.io/FORT-validator/)
  - Packages available for Debian/Ubuntu, RHEL/CentOS & FreeBSD
  - (Can also be built from source)
RPKI Validator Caches (2)

- **RPKI-client**
  - [https://www.rpki-client.org/](https://www.rpki-client.org/)
  - [https://tracker.debian.org/pkg/rpki-client](https://tracker.debian.org/pkg/rpki-client)
  - RPKI repository query system (output for OpenBGPD, BIRD, json)
  - For OpenBSD, with ports for Debian/Ubuntu, RHEL/CentOS, FreeBSD, macOS

- **StayRTR**
  - [https://github.com/bgp/stayrtr](https://github.com/bgp/stayrtr)
  - [https://tracker.debian.org/pkg/stayrtr](https://tracker.debian.org/pkg/stayrtr)
  - RPKI to Router protocol implementation (input JSON formatted VRP exports)
  - (hard fork of Cloudflare GoRTR)
  - Works on anything Go runs on (?)

- **Note:**
  - RPKI-client and StayRTR are used together
RPKI Validator Caches (3)

- **RPKI-Prover**
  - [https://github.com/lolepezy/rpki-prover](https://github.com/lolepezy/rpki-prover)

- **rpstir2**
  - [https://github.com/bgpsecurity/rpstir2](https://github.com/bgpsecurity/rpstir2)

- **No longer maintained – out of date, do NOT use:**
  - Dragon Research Labs “rcynic”
  - Cloudflare validator (OctoRPKI/GoRTR)
    - StayRTR is a fork of GoRTR
  - RIPE NCC validator
    - Version 2 and 3
Installing a validator

- Three validators are widely used
  - Routinator 3000
  - FORT
  - RPKI-client/StayRTR
- Listed in order of ease of installation 😊
- For installation details on Ubuntu 22.04
  - https://bgp4all.com/pfs/hints/rpki
Installing a validator – Routinator

- If using Ubuntu/Debian, then simply use the package manager, as described:

- In summary:
  - Get the NLnetLabs public key
  - Add the repo to the sources lists
  - Install routinator
  - Initialise
  - Run

```
philip@rpki:~$ sudo apt install routinator
Reading package lists... Done
Building package lists... Done

philip@rpki:~$ wget -4 -qO- https://packages.nlnetlabs.nl/optkey.asc | sudo apt-key add -
philip@rpki:~$ cat /etc/apt/sources.list.d/routinator-bionic.list
deb [arch=amd64] https://packages.nlnetlabs.nl/linux/ubuntu/ bionic main
philip@rpki:~$ sudo apt-get update
philip@rpki:~$ sudo apt-get install routinator
Reading package lists... Done
Building package lists... Done

philip@rpki:~$ sudo routinator-init --accept-origin-rpki
Creating local repository directory /var/lib/routinator/rpki-cache
Installed 5 TAs in /var/lib/routinator/tals
..."
Routinator 3000 web interface

- User interface of Routinator accessed by enabling http option in the server configuration
  - Listens on port 8323

/etc/routinator/routinator.conf
Installing a validator – FORT

- Easiest is to download one of the packages available
  - Described at https://nicmx.github.io/FORT-validator/installation.html
  - Example for Ubuntu 20.04:

```bash
philip@fort-$ wget https://github.com/NICMX/FORT-validator/releases/download/1.5.3/fort_1.5.3-1_amd64.deb
--2022-01-20 13:00:49- https://github.com/NICMX/FORT-validator/releases/download/1.5.3/fort_1.5.3-1_amd64.deb
Resolving github.com (github.com)... 
<snip>
HTTP request sent, awaiting response... 200 OK
Length: 21436 (209K) [application/octet-stream]
Saving to: 'fort_1.5.3-1_amd64.deb'

41%[--------------------------] 209.12k  (29.99s)

philip@fort-$ sudo apt install ./fort_1.5.3-1_amd64.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note: selecting 'fort' instead of 'fort_1.5.3-1_amd64.deb'.
The following additional packages will be installed:
  libjsoncpp4
The following NEW packages will be installed:
  fort
  libjsoncpp4
  The following NEW packages will be installed:
  fort
  libjsoncpp4
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded
Need to get 28.9 kB/243 kB of archives.
After this operation, 795 kB of additional disk space will be
Do you want to continue? [Y/n] y
Get:1 /home/philip/fort/1.5.3-1_amd64.deb fort_1.5.3-1_amd64.deb [28.9kB]
Fetched 28.9 kB in 1s (30.0 kB/s)
philip@fort-$
```

- Note the automatic creation of the systemd entry
- The configuration file is `/etc/fort/config.json` – set the listening port here (323 by default)
Running FORT

- **Other notes:**
  - Need to refresh the TALs before starting
  - Need to make sure that `/var/lib/fort` is owned by the `fort` user
  - Otherwise FORT will crash on startup with these errors because it cannot write there:

```
Jan 20 13:33:22 fort[5768]: Stack trace:
Jan 20 13:33:22 fort[5768]: /usr/bin/fort(print_stack_trace=0xb37) [0x55e4d7e27fc8]
Jan 20 13:33:22 fort[5768]: /usr/bin/fort(handle_flags_config=0xb388) [0x55e4d7e27fc8]
Jan 20 13:33:22 fort[5768]: /usr/bin/fort(main=0xb66) [0x55e4d7e232c6]
Jan 20 13:33:22 fort[5768]: /lib/x86_64-linux-gnu/libc.so.6(_libc_start.main)
Jan 20 13:33:22 fort[5768]: /usr/bin/fort(start=0xb2a) [0x55e4d7e233fa]
Jan 20 13:33:22 fort[5768]: (End of stack trace)
Jan 20 13:33:22 fort[5768]:fort.service: Main process exited, code=exited, signal=15
Jan 20 13:33:22 fort[5768]:fort.service: Failed with result 'exit-code'.
```
Installing rpki-client (1)

- rpki-client is packaged from Ubuntu 22.04 onwards but is an old version
- Best built from scratch to get the latest
  - Easiest is to build from the Git repository:
- Note the instructions to get the environment ready:
  - You will need automake, autoconf, git, libtool, and libexpat-dev to be installed first – use the package manager
  - LibreSSL tls is also needed – this is part of OpenBSD but the source will compile on Linux
  - Get latest LibreSSL:
    - [https://ftp.openbsd.org/pub/OpenBSD/LibreSSL/](https://ftp.openbsd.org/pub/OpenBSD/LibreSSL/)
  - Unpack and then run:
    ```
    ./configure --enable-libtls-only
    make
    make install
    ```
  - Which will build and install the libtls that rpki-client needs
Installing rpki-client (2)

- With the environment ready
  - Run `./autogen.sh` inside the rpki-client distribution
  - Then run
    ```
    ./configure --with-tal-dir=/etc/rpki \
    --with-base-dir=/var/lib/rpki-client \
    --with-output-dir=/var/db/rpki-client
    ```
  - And finally build the client by running `make`
Running rpki-client

- Before we install the client we need to add the specific user and group that the client will use:

  ```bash
  sudo groupadd _rpki-client
  sudo useradd -g _rpki-client -s /sbin/nologin -d /nonexistent -c "rpki-client user" _rpki-client
  ```

- And then we can run:

  ```bash
  sudo make install
  ```

  Which will install the client in /usr/local/sbin and the 4 TALs in /etc/rpki, as well as create the cache and output directories needed.

- ARIN TAL requires users to read the disclaimer first:

  ```bash
  https://www.arin.net/resources/manage/rpki/arin.tal
  ```

- Now the client can be run (at the command-line, no daemon)

  ```bash
  philip@validator:~$ sudo /usr/local/sbin/rpki-client
  rpki-client: https://rrdp.krill.cloud/notification.xml: connect: Connection refused
  rpki-client: Error retrieving ca.rg.net: 404 NOT FOUND
  rpki-client: https://rrdp.taa.a.eu/rrdp/notification.xml: connect: Connection refused
  rpki-client: https://rrdp.taa.a.eu/rrdp/notification.xml: connect: Connection refused
  ```

- Client authors recommend running the client hourly by cron

  ```bash
  ```

  See [https://man.openbsd.org/rpki-client](https://man.openbsd.org/rpki-client) for more information about output options.
Installing StayRTR

- StayRTR is packaged from Ubuntu 22.04 onwards but is an old version
- Best built from scratch
  - Easiest is to build from the Git repository: https://github.com/bgp/stayrtr
- You will also need a working Go environment
  - The Go site has more information: https://go.dev/doc/install
- And then you can build StayRTR:
  ```
  cd stayrtr
  make build-stayrtr
  ```
- Put resultant binary into /usr/local/bin
Running StayRTR

- StayRTR has lots of options
  - The ones we need are:
    -bind string
      Bind address (default ":8282")
    -cache string
      URL of the cached JSON data (default "https://console.rpki-client.org/vrps.json")

- We have set up our rpki-client to save the data in /var/db/rpki-client
  - Run the client like this:
    /usr/local/bin/stayrtr -bind :3323 -cache /var/db/rpki-client/json
RP Cache Deployment

Network Operator design advice:
- Deploy at least two Validator Caches
- Geographically diverse
- Two different implementations
  - For software independence
- Implement on a Linux container so that the container can be moved between different server clusters as required
- Configure validator to listen on both IPv4 and IPv6
  - Configure routers with both IPv4 and IPv6 validator connections
- Securing the validator: Only permit routers running EBGP to have access to the validators
Consider two different validator cache implementations

- Gives software independence
- What happens if the different cache implementations contain different VRPs?

- Scenario 1:
  - Cache 1: route X is valid
  - Cache 2: route X is invalid

- Scenario 2:
  - Cache 1: route X is valid
  - Cache 2: route X is NotFound

Answer: depends on router vendor implementation?!
Configure Router to Use Cache: Cisco IOS

- Point router to the local RPKI cache
  - Server listens on port 3323
  - Cache refreshed every 60 minutes (RFC8210 recommendation)
  - Example:

    ```
    router bgp 64512
    bgp rpki server tcp 10.0.0.3 port 3323 refresh 3600
    ```

- Once the router’s RPKI table is populated, router indicates validation state in the BGP table
Cisco IOS status commands

- `show ip bgp rpki servers`
  - Displays the connection status to the RPKI caches

- `show ip bgp rpki table`
  - Shows the VRPs (validated ROA payloads)

- `show ip bgp`
  - Shows the BGP table with status indication next to the prefix

- `show ip bgp | i ^V`
  - Shows the status “valid” prefixes in the BGP table
Configure Router to Use Cache: JunOS

1. Connect to validation cache:

   routing-options {
       validation {
           group ISP {
               session <ip address validator> {
                   refresh-time 600;
                   hold-time 3600;
                   preference 1;
                   port <port>;
                   local-address <router loopback>;
               }
           }
       }
   }

   (using same parameters as for the Cisco IOS example)
Configure Router to Use Cache: JunOS

2. Configure validation policies:

```plaintext
policy-options {
    policy-statement RPKI-validation {
        term VALID {
            from {
                protocol bgp;
                validation-database valid;
            }
            then {
                validation-state valid;
                next policy;
            }
        }
        term INVALID {
            from {
                protocol bgp;
                validation-database invalid;
            }
            then {
                validation-state invalid;
                next policy;
            }
        }
        term UNKNOWN {
            from {
                protocol bgp;
                validation-database unknown;
            }
            then {
                validation-state unknown;
                next policy;
            }
        }
    }
}
(continued)...

To drop invalid prefixes, replace the next policy with reject
```
3. **Apply policy to eBGP session:**

```plaintext
protocols {
    bgp {
        group EBGP {
            type external;
            local-address 10.0.1.1;
            neighbor 10.1.15.1 {
                description "ISP Upstream";
                import [ RPKI-validation Upstream-in ];
                export LocalAS-out;
                peer-as 64511;
            }
        }
    }
}
```

- Note that policy options *Upstream-in* and *LocalAS-out* are the typical inbound and outbound filters needed for an eBGP session.
JunOS status commands

- `show validation session detail`
  - Display the details of the connection to the RPKI caches

- `show validation replication database`
  - Shows the VRPs (validated ROA payloads)

- `show route protocol bgp`
  - Shows the BGP table with status indication next to the prefix
    - `show route protocol bgp validation-state valid`
  - Shows the status “valid” prefixes in the BGP table
Configure Router to Use Cache: FRrouting

- Point router to the local RPKI cache
  - Server listens on port 3323
  - Cache refreshed every 60 minutes (RFC8210 recommendation)
  - Example:

```
rpki
  rpki polling_period 3600
  rpki cache 10.0.0.3 3323 preference 1
  rpki cache 10.0.1.2 3323 preference 2
exit
```

- Two caches specified for redundancy
**FRrouting status commands**

- `show rpki cache-connection`
  - Displays the connection status to the RPKI caches

- `show rpki prefix-table`
  - Shows the VRPs (validated ROA payloads)

- `show ip bgp`
  - Shows the BGP table

- `show ip bgp rpki valid`
  - Shows the status “valid” prefixes in the BGP table
  - (There are also options for “invalid” and “notfound”)


Configure Router to Use Cache: BIRD v2

- Point BIRD to the local RPKI cache
  - Server listens on port 3323
  - Cache refreshed every 60 minutes (RFC8210 recommendation)
  - Two caches specified for redundancy

```
roa4 table r4;
roa6 table r6;

protocol rpki validator1 {
  roa4 { table r4; };
  roa6 { table r6; };
  remote 10.0.0.3 port 3323;
  retry 300;
}

protocol rpki validator2 {
  roa4 { table r4; };
  roa6 { table r6; };
  remote 10.0.1.2 port 3323;
  retry 300;
}
```
## BIRD v2 status commands

- **show protocols validator1**
  - Displays the connection status to the RPKI cache “validator1”

- **show route table r4**
  - Shows the IPv4 VRPs (validated ROA payloads)

- **show route table r6**
  - Shows the IPv6 VRPs (validated ROA payloads)

- **show route protocol <name>**
  - Shows the BGP table (but has no validation info)
Implementation notes

- Cisco IOS/IOS-XE
  - Invalid prefixes are dropped by default
    - The operator does not need to define a policy based on validation state
  - Prefixes originated locally into IBGP are automatically marked as Valid
    - There is no check against the cached validation table
    - Allows operator to originate non-signed address blocks or other entity address space inside their own IBGP

- JunOS
  - Complete separation between validation table and what happens in BGP
    - There has to be a specific policy statement for any action based on validation state
Implementation notes

- **Cisco IOS/IOS-XE/IOS-XR**
  - Every VRP change causes a route-refresh with its BGP neighbours
    - Even though VRP change only affects valid/invalid/notfound status
  - Big impact for BGP sessions carrying a large or the full BGP table
    - Especially for BGP peers with weak control planes!
  - Transit providers need to be cautious:
    - BGP customer doing ROV on Cisco router will cause significant impact on the Access Router CPU
  - Cisco’s recommended workaround:
    - Turn on “Soft Reconfiguration”
    - Which has memory implications, and blocks access to the route refresh CLI
  - **Summary:** think carefully about using Cisco routers for Route Origin Validation
Implementation notes

- **Other router implementations**
  - Most modern implementations save the incoming BGP table prior to policy application (ADJ-RIB-IN)
  - Changes in VRPs are applied to this stored BGP table
  - Similar behaviour to Cisco’s soft-reconfiguration

- **NB: It’s important not to rely on Route Refresh to implement VRP changes**
  - More and more frequent changes cause more and more refresh requests to peers, consuming peer CPU resources – potentially a denial-of-service attack on the peer
  - Recommended reading:
    - [https://www.rfc-editor.org/info/rfc9324](https://www.rfc-editor.org/info/rfc9324)
Implementation notes

- What happens when router cannot contact any validator cache?
  - Cisco IOS/IOS-XE – empties the VRP table within 5 minutes
  - Juniper & Nokia – keeps VRPs until their preconfigured expiry (default 60 minutes)
  - Other vendors – behaviour untested

- Design advice:
  - It is important to ensure that EBGP speaking routers can always remain connected to a validator cache
    - Minimum of two independent caches recommended!
route-views>sh ip bgp rpki ser
BGP SOVC neighbor is 128.223.157.83/3323 connected to port 3323
Flags 64, Refresh time is 300, Serial number is 542, Session ID is 11962
InQ has 0 messages, OutQ has 0 messages, formatted msg 3408
Session IO flags 3, Session flags 4008
Neighbor Statistics:
  Prefixes 669770
  Connection attempts: 1170
  Connection failures: 1074
  Errors sent: 0
  Errors received: 2

Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Minimum incoming TTL 0, Outgoing TTL 255
Local host: 128.223.51.103, Local port: 64209
Foreign host: 128.223.157.83, Foreign port: 3323
Connection tableid (VRF): 0
Maximum output segment queue size: 50

Courtesy of RouteViews
Check Server

philip@DREN-THIMPHU-BR> show validation session detail
Session 103.197.176.141, State: up, Session index: 2
  Group: DrukREN, Preference: 100
  Local IPv4 address: 103.197.176.5, Port: 3323
  Refresh time: 600s
  Hold time: 1800s
  Record Life time: 3600s
  Serial (Full Update): 0
  Serial (Incremental Update): 1
    Session flaps: 1
    Session uptime: 00:19:11
    Last PDU received: 00:00:34
    IPv4 prefix count: 94329
    IPv6 prefix count: 15992

Courtesy of DrukREN, Bhutan
RPKI Table (IPv4) – May 2024 – RouteViews

<table>
<thead>
<tr>
<th>Network</th>
<th>Maxlen</th>
<th>Origin-AS</th>
<th>Source</th>
<th>Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/24</td>
<td>24</td>
<td>13335</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.4.0/24</td>
<td>24</td>
<td>38803</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.4.0/22</td>
<td>22</td>
<td>38803</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.5.0/24</td>
<td>24</td>
<td>38803</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.6.0/24</td>
<td>24</td>
<td>38803</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.7.0/24</td>
<td>24</td>
<td>38803</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.0.64.0/18</td>
<td>18</td>
<td>18144</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.1.1.0/24</td>
<td>24</td>
<td>13335</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.1.4.0/22</td>
<td>22</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.1.16.0/20</td>
<td>20</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.2.9.0/24</td>
<td>24</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.2.10.0/24</td>
<td>24</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.2.11.0/24</td>
<td>24</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.2.12.0/22</td>
<td>22</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.3.0.0/16</td>
<td>16</td>
<td>4134</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>1.6.0.0/22</td>
<td>24</td>
<td>9583</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
</tbody>
</table>
### RPKI Table (IPv6) – May 2024 – RouteViews

222696 BGP sovc network entries using 40976064 bytes of memory
229077 BGP sovc record entries using 7330464 bytes of memory

<table>
<thead>
<tr>
<th>Network</th>
<th>Maxlen</th>
<th>Origin-AS</th>
<th>Source</th>
<th>Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:200::/32</td>
<td>32</td>
<td>2500</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:1BA::/48</td>
<td>48</td>
<td>24047</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:900::/40</td>
<td>40</td>
<td>7660</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:E00::/40</td>
<td>40</td>
<td>4690</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:8000::/35</td>
<td>35</td>
<td>4690</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:C000::/35</td>
<td>35</td>
<td>23634</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:200:E000::/35</td>
<td>35</td>
<td>7660</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:201::/32</td>
<td>32</td>
<td>2914</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:202::/31</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:204::/30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:209::/32</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:20A::/31</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:20C::/30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:210::/29</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
<tr>
<td>2001:218::/32</td>
<td>32</td>
<td>2914</td>
<td>0</td>
<td>128.223.157.83/3323</td>
</tr>
</tbody>
</table>
### BGP Table (IPv4)

**RPKI validation codes: V valid, I invalid, N Not found**

<table>
<thead>
<tr>
<th>Network</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>V*&gt; 1.0.0.0/24</td>
<td>0</td>
<td>17660 6453 4755 13335 i</td>
<td></td>
</tr>
<tr>
<td>V*&gt; 1.0.4.0/22</td>
<td>0</td>
<td>17660 7545 2764 38803 i</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V*&gt; 1.9.0.0/16</td>
<td>0</td>
<td>17660 6939 4788 i</td>
<td></td>
</tr>
<tr>
<td>V*&gt; 1.9.250.0/24</td>
<td>0</td>
<td>17660 2914 1299 6939 6939 4788 i</td>
<td></td>
</tr>
<tr>
<td>V*&gt; 1.10.10.0/24</td>
<td>0</td>
<td>17660 2914 64049 55836 9885 142501 148000 i</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V*&gt; 1.7.228.0/23</td>
<td>0</td>
<td>17660 6453 9583 i</td>
<td></td>
</tr>
<tr>
<td>I*&gt; 1.7.228.0/24</td>
<td>0</td>
<td>17660 6453 4755 9583 137130 i</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V*&gt; 23.252.75.0/24</td>
<td>0</td>
<td>17660 6453 6939 i</td>
<td></td>
</tr>
<tr>
<td>V*&gt; 23.252.76.0/24</td>
<td>0</td>
<td>17660 6453 6939 i</td>
<td></td>
</tr>
<tr>
<td>I*&gt; 23.252.77.0/24</td>
<td>0</td>
<td>17660 6453 3257 i</td>
<td></td>
</tr>
<tr>
<td>I*&gt; 23.252.78.0/24</td>
<td>0</td>
<td>17660 6453 3257 i</td>
<td></td>
</tr>
<tr>
<td>V*&gt; 23.252.79.0/24</td>
<td>0</td>
<td>17660 6453 6939 i</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Courtesy of RouteViews
**BGP Table (IPv6)**

RPKI validation codes: V valid, I invalid, N Not found

<table>
<thead>
<tr>
<th>Network</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>N*&gt; 2001::/32</td>
<td>0</td>
<td>17660</td>
<td>2914 6939 i</td>
</tr>
<tr>
<td>N*&gt; 2001:4:112::/48</td>
<td>0</td>
<td>17660</td>
<td>36236 112 ii</td>
</tr>
<tr>
<td>V*&gt; 2001:200::/32</td>
<td>0</td>
<td>17660</td>
<td>2914 2500 2500 i</td>
</tr>
<tr>
<td>V*&gt; 2001:200:900::/40</td>
<td>0</td>
<td>17660</td>
<td>2914 17676 2500 2500 7660 i</td>
</tr>
<tr>
<td>V*&gt; 2001:200:e00::/40</td>
<td>0</td>
<td>17660</td>
<td>2914 17676 2500 2500 4690 i</td>
</tr>
<tr>
<td>V*&gt; 2001:200:c000::/35</td>
<td>0</td>
<td>17660</td>
<td>6939 23634 23634 ii</td>
</tr>
<tr>
<td>V*&gt; 2001:5a0::/32</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>I*&gt; 2001:5a0:3f06::/48</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>V*&gt; 2001:5a0:4402::/48</td>
<td>0</td>
<td>17660</td>
<td>2914 20940 i</td>
</tr>
<tr>
<td>I*&gt; 2001:5a0:4604::/48</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>I*&gt; 2001:5a0:4e01::/48</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>I*&gt; 2001:5a0:9000::/36</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>V*&gt; 2001:5a0:9000::/38</td>
<td>0</td>
<td>17660</td>
<td>6453 i</td>
</tr>
<tr>
<td>V*&gt; 2001:5a0:9001::/48</td>
<td>0</td>
<td>17660</td>
<td>6453 7029 i</td>
</tr>
</tbody>
</table>

Courtesy of RouteViews
RPKI BGP State: Valid

route-views>sh bgp ipv6 unicast 2001:240::/32
BGP routing table entry for 2001:240::/32, version 95472891
Paths: (13 available, best #10, table default)
   Not advertised to any peer
   Refresh Epoch 1
   20912 6939 2497
      2001:40D0::1E from 2001:40D0::1E (77.39.192.1)
      Origin IGP, localpref 100, valid, external
      path 7F16801AE960 RPKI State valid
      rx pathid: 0, tx pathid: 0
   Refresh Epoch 1
route-views>sh bgp ipv6 unicast 2001:5a0:3f06::/48
BGP routing table entry for 2001:5a0:3f06::/48, version 98192653
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  20912 49367 6762 6453
    2001:40d0::1e from 2001:40d0::1e (77.39.192.1)
      Origin IGP, localpref 100, valid, external, best
      path 7f1561494a90 RPKI State invalid
      rx pathid: 0, tx pathid: 0x0
route-views>sh bgp ipv6 unicast 2001::/32
BGP routing table entry for 2001::/32, version 95354292
Paths: (11 available, best #6, table default)
   Not advertised to any peer
   Refresh Epoch 1
   20912 6939
      2001:40D0::1E from 2001:40D0::1E (77.39.192.1)
      Origin IGP, localpref 100, valid, external
      path 7F16801B0F60  RPKI State not found
      rx pathid: 0, tx pathid: 0
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
Deploying RPKI

Deploying and using RPKI
Network operators can make decisions based on RPKI state:
- Invalid – discard the prefix – please do this now!
- NotFound – let it through (maybe low local preference)
- Valid – let it through (high local preference)

Some operators even considering making “Not Found” a discard event
- But then Internet IPv4 BGP table would shrink to about 450000 prefixes and the IPv6 BGP table would shrink to about 950000 prefixes!
Using RPKI

- **Invalid** means discarding the prefix
  - Which means it does not go into the BGP RIB or the FIB
  - And that means routing falls back to the covering aggregate
    - (Subnet might be a hijack, so the covering aggregate will be the correct path)
  - In the absence of a covering aggregate, the default route will be used

- Running ROV in an AS with a default route?
  - Invalids dropped, so it is likely the default will be used instead
  - Either move into default free zone (full tables), otherwise implementing ROV is more an academic/educational exercise
Using RPKI

Should invalid routes be routed to the Null/Discard interface rather than just dropped?

- If this is done, then access to that route is totally blocked, even if the covering aggregate offers the legitimate path to the destination
  - Which means the hijack is inadvertently partially successful because access to the legitimate destination is prevented

- Answer: NO, please don’t do this
Deploying RPKI within an AS

- For fully supported Route Origin Validation across the network:
  - All EBGP speaking routers need to talk with a validator
    - Supporting ROV means dropping invalids as they arrive in the network
    - EBGP speaking routers are part of the operator IBGP mesh
  - IBGP speaking routers do not need to talk with a validator
    - Only valid and NotFound prefixes will be distributed from the EBGP speaking routers
    - The validation table is not distributed from router to router

- Remember:
  - Cisco IOS/IOS-XE drops invalids by default – to allow invalids to be distributed by IBGP, use the per address-family command:

    ```bash
cgrep bestpath prefix-validate allow-invalid
```
RFC8097 describes the propagation of validation state between iBGP speakers

- Defines an opaque extended BGP community

<table>
<thead>
<tr>
<th>Extended Community</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4300:0:0</td>
<td>Valid</td>
</tr>
<tr>
<td>0x4300:0:1</td>
<td>NotFound</td>
</tr>
<tr>
<td>0x4300:0:2</td>
<td>Invalid</td>
</tr>
</tbody>
</table>

- These extended communities can be used in IBGP to allow distribution of validation state along with the prefix if desired
- On Cisco IOS/IOS-XE:
  ```
  neighbor x.x.x.x announce rpki state
  ```
- For JunOS, policy needs to be explicitly configured
Propagating validation state

- There are two important caveats when propagating validation state:
  - Interoperability – is the defined opaque extended community supported on all vendor equipment in a multi-vendor network?
    - Until recently JunOS would not allow the required opaque extended communities to be configured at the command line
  - Cisco IOS/IOS-XE behaviour:
    - Adds a step to the best path selection algorithm: checks validation state (valid preferred over not found) before checking local preference
      - This cannot be turned off 😞
JunOS: opaque extended community

- Supported only in more recent JunOS releases
  - Fixed from 17.4R3, 18.2R3, 18.4R2...

```plaintext
policy-options {
    community RPKI-VALID members 0x4300:0:0;
    community RPKI-UNKNOWN members 0x4300:0:1;
    community RPKI-INVALID members 0x4300:0:2;
}
```
JunOS: opaque extended community

- And we can now set policy to detect these communities being sent from Cisco IOS/IOS-XE routers
  - Under “policy-options”:
Propagating validation state: Cisco IOS

- Cisco IOS/IOS-XE behaviour – example:
  - Prefix learned via two paths via two separate EBGP speaking routers
  - Prefix and validation state distributed by IBGP to core router (route reflector):

<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>Metric</th>
<th>LocPrf</th>
<th>Weight</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>V*&gt;i 61.45.249.0/24</td>
<td>100.68.1.1</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>121 20 135534 i</td>
</tr>
<tr>
<td>i 100.68.1.3</td>
<td>0</td>
<td>200</td>
<td>0</td>
<td>20</td>
<td>135534 i</td>
</tr>
<tr>
<td>V*&gt;i 61.45.250.0/24</td>
<td>100.68.1.1</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>121 30 135535 i</td>
</tr>
<tr>
<td>i 100.68.1.3</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>30</td>
<td>135535 i</td>
</tr>
<tr>
<td>V*&gt;i 61.45.251.0/24</td>
<td>100.68.1.1</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>121 122 40 135536 i</td>
</tr>
<tr>
<td>i 100.68.1.3</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>40</td>
<td>135536 i</td>
</tr>
</tbody>
</table>

- One EBGP speaking router talks with validator
- The other EBGP speaking router does not (due to error or design)
- Core router best path selection prefers valid path over not found even if the latter has higher local preference
Looking at the path detail:

BGP routing table entry for 61.45.249.0/24, version 32
BGP Bestpath: deterministic-med
Paths: (2 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  121 20 135534, (Received from a RR-client)
    100.68.1.1 (metric 2) from 100.68.1.1 (100.68.1.1)
      Origin IGP, metric 0, localpref 50, valid, internal, best
      Extended Community: 0x4300:0:0
      path 67A585D0 RPKI State valid
  Refresh Epoch 1
  20 135534, (Received from a RR-client)
    100.68.1.3 (metric 2) from 100.68.1.3 (100.68.1.3)
      Origin IGP, metric 0, localpref 200, valid, internal
      Community: 10:1100
      Extended Community: 0x4300:0:1
      path 67A58918 RPKI State not found

Note best path
Propagating validation state

- Consider **carefully** if this is desired
- Current standard practice is to:
  - EBGP speaking routers have session with two diverse/redundant validators
  - Check validation state on EBGP speaking routers
  - Drop invalids on EBGP speaking routers
  - Distribute remaining prefixes by IBGP
  - Avoid propagating validation state (at least in Cisco IOS)
    - or-
  - Make sure that EBGP speaking routers never lose their connectivity to validators
RPKI Summary

- All AS operators must consider deploying:
  - **Signing ROAs**
  - **Dropping Invalids** (ROV)
- An important step to securing the routing system
- Doesn’t secure the path, but that’s the next important hurdle to cross
- With origin validation, the opportunities for malicious or accidental mis-origination are considerably reduced
- FAQ:
  - [https://nlnetlabs.nl/projects/rpki/faq/](https://nlnetlabs.nl/projects/rpki/faq/)
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
RPKI Deployment Status
RPKI Deployment Status

- NIST keeps track of deployment status for research purposes:
  - [https://rpki-monitor.antd.nist.gov/](https://rpki-monitor.antd.nist.gov/)

- IIJ Labs RPKI statistics:
  - [https://ihr.iijlab.net/ihr/en-us/rov](https://ihr.iijlab.net/ihr/en-us/rov)

- RIPE NCC statistics:
  - [https://certification-stats.ripe.net/](https://certification-stats.ripe.net/)

- NSRC ROA status:
  - Routinator Validator running at NSRC
  - [https://routinator.nsrc.org/](https://routinator.nsrc.org/)
Number of ROAs

https://certification-stats.ripe.net/
IPv4 Address Space in ROAs (/24s)

https://certification-stats.ripe.net/
IPv6 Address Space in ROAs (/48s)

https://certification-stats.ripe.net/
### Route Origin Validation

3-day report ending on 21 May 2024

**IIJ Labs RPKI Statistics**

https://ihr.iijlab.net/ihr/

<table>
<thead>
<tr>
<th>Country</th>
<th>Origin ASN</th>
<th>Prefix</th>
<th>Route</th>
<th>RPKI</th>
<th>IRR</th>
<th>Status</th>
<th>Origin ASN</th>
<th>Visibility</th>
<th>Main Transits</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>AS6939</td>
<td>45.12.83.0/24</td>
<td></td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>66.8%</td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>AS132839</td>
<td>156.255.216.0/24</td>
<td>IOCC Limited</td>
<td>✗ Invalid</td>
<td>✗ Valid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>65.2%</td>
<td></td>
</tr>
<tr>
<td>ZZ</td>
<td>AS13335</td>
<td>103.21.244.0/24</td>
<td>Cloudflare Hong Kong, LLC 101 Townsend Street</td>
<td>✗ Invalid</td>
<td>✗ Invalid (more specific)</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>63.6%</td>
<td>AS13335</td>
</tr>
<tr>
<td>HK</td>
<td>AS13335</td>
<td>156.254.32.0/19</td>
<td>CMIP Transit</td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>63.6%</td>
<td>AS13335</td>
</tr>
<tr>
<td>US</td>
<td>AS13335</td>
<td>68.67.65.0/24</td>
<td>SBA EDGE COLLOCATION CUSTOMERS (former GDRack) - AreIon announce</td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>63.6%</td>
<td></td>
</tr>
<tr>
<td>VG</td>
<td>AS209242</td>
<td>194.40.241.0/24</td>
<td></td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>61.5%</td>
<td>AS13335</td>
</tr>
<tr>
<td>ZA</td>
<td>AS209242</td>
<td>154.16.94.0/24</td>
<td>AS209242</td>
<td>✗ Invalid</td>
<td>✗ Valid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>59.9%</td>
<td>AS13335</td>
</tr>
<tr>
<td>ZZ</td>
<td>AS13335</td>
<td>2606:4700:7000::/48</td>
<td>101 Townsend Street, San Francisco, California 94107, US</td>
<td>✗ Invalid</td>
<td>✗ Invalid (more specific)</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>59.6%</td>
<td>AS138195, AS158830, AS17819, AS18390, AS38195, AS33386, AS64461, AS7473</td>
</tr>
<tr>
<td>AU</td>
<td>AS36040</td>
<td>202.172.96.0/19</td>
<td>Proxy route object registered by AS38196</td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>56.7%</td>
<td>AS36823, AS9304, AS6762, AS7473, AS64801</td>
</tr>
<tr>
<td>KH</td>
<td>AS54994</td>
<td>45.64.127.0/24</td>
<td>Proxy-registered route object</td>
<td>✗ Invalid</td>
<td>✗ Invalid</td>
<td>✗ assigned</td>
<td>✗ assigned</td>
<td>56.7%</td>
<td>AS64801</td>
</tr>
</tbody>
</table>
**Route Origin Validation**

3-day report ending on 21 May 2024

<table>
<thead>
<tr>
<th>ASN</th>
<th>Name</th>
<th>RPKI invalid</th>
<th>RPKI invalid (more specific)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS18181</td>
<td>RELIANCE COMMUNICATIONS-IN Reliance Communications Ltd DAIK MUMBAI, IN</td>
<td>2</td>
<td>359</td>
<td>361</td>
</tr>
<tr>
<td>AS7029</td>
<td>WINDSTREAM, US</td>
<td>1</td>
<td>151</td>
<td>152</td>
</tr>
<tr>
<td>AS4804</td>
<td>MPX-AS Microplex PTY LTD, AU</td>
<td>0</td>
<td>115</td>
<td>115</td>
</tr>
<tr>
<td>AS12389</td>
<td>ROSTELECOM-AS PJSC Rostelecom, RU</td>
<td>83</td>
<td>7</td>
<td>90</td>
</tr>
<tr>
<td>AS39891</td>
<td>ALJAWWALSTC-AS Saudi Telecom Company JSC, SA</td>
<td>80</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>AS21491</td>
<td>UGANDA-TELECOM Uganda Telecom, UG</td>
<td>0</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>AS58224</td>
<td>TCI Iran Telecommunication Company PJSC, IR</td>
<td>0</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>AS43940</td>
<td>MTEL AS Drustvo za telekomunikacije 'MTEL' DOO, ME</td>
<td>0</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>AS7713</td>
<td>TELKOMNET-AS-AP PT Telekomunikasi Indonesia, ID</td>
<td>1</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>AS45090</td>
<td>TENCENTNET-AP Shenzhen Tencent Computer Systems Company Limited, CN</td>
<td>0</td>
<td>36</td>
<td>36</td>
</tr>
</tbody>
</table>
## Route Origin Validation

3-day report ending on 21 May 2024

### RPKI invalid

<table>
<thead>
<tr>
<th>ASN</th>
<th>Name</th>
<th>RPKI invalid</th>
<th>RPKI invalid (more specific)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS6762</td>
<td>SEABONE-NET TELECOM ITALIA SPARKLE S.p.A., IT</td>
<td>680</td>
<td>1484</td>
<td>2164</td>
</tr>
<tr>
<td>AS6453</td>
<td>AS6453, US</td>
<td>145</td>
<td>348</td>
<td>493</td>
</tr>
<tr>
<td>AS7473</td>
<td>SINGTEL-AS-AP Singapore Telecommunications Ltd, SG</td>
<td>98</td>
<td>298</td>
<td>396</td>
</tr>
<tr>
<td>AS15412</td>
<td>FLAG-AS FLAG TELECOM UK LIMITED, GB</td>
<td>18</td>
<td>377</td>
<td>395</td>
</tr>
<tr>
<td>AS3396</td>
<td>LEVEL3, US</td>
<td>160</td>
<td>227</td>
<td>387</td>
</tr>
<tr>
<td>AS6461</td>
<td>ZAYO-6461, US</td>
<td>66</td>
<td>271</td>
<td>337</td>
</tr>
<tr>
<td>AS9498</td>
<td>BBIL-AP BHARTI Airtel Ltd., IN</td>
<td>72</td>
<td>170</td>
<td>242</td>
</tr>
<tr>
<td>AS49666</td>
<td>TIC-GW-AS Telecommunication Infrastructure Company, IR</td>
<td>24</td>
<td>110</td>
<td>134</td>
</tr>
<tr>
<td>AS4735</td>
<td>TATACOMM-AS TATA Communications formerly VSNL is Leading ISP, IN</td>
<td>56</td>
<td>49</td>
<td>105</td>
</tr>
<tr>
<td>AS29386</td>
<td>STC-IGW-AS Saudi Telecom Company JSC, SA</td>
<td>87</td>
<td>3</td>
<td>90</td>
</tr>
</tbody>
</table>

[https://ihr.iiilab.net/ihr/]
Major Operators deploying RPKI and ROV

- Arelion

<table>
<thead>
<tr>
<th>aut-num:</th>
<th>AS1299</th>
</tr>
</thead>
<tbody>
<tr>
<td>org:</td>
<td>ORG-TCA32-RIPE</td>
</tr>
<tr>
<td>as-name:</td>
<td>TWELVE99</td>
</tr>
<tr>
<td>descr:</td>
<td>Arelion, f/k/a Telia Carrier</td>
</tr>
<tr>
<td>snip:</td>
<td></td>
</tr>
<tr>
<td>remarks:</td>
<td>AS1299 is matching RPKI validation state and reject invalid prefixes from peers and customers.</td>
</tr>
<tr>
<td>remarks:</td>
<td>Please review your registered ROAs to reduce number of invalid prefixes.</td>
</tr>
</tbody>
</table>
Major Operators deploying RPKI and ROV

- More and more operators are deploying RPKI and ROV
- Not just transit providers!
- But also:
  - Content providers
  - IXPs
  - R&E networks
  - Access providers

- Arelion
- NTT
- Lumen
- HE
- GTT
- Workonline
- SEACOM
- Cloudflare
- AMS-IX
- LINX
- DE-CIX
- Terrehost
- Vocus
- Telstra
- REANNZ
- Cogent
- GR-IX
- Swisscom
- Netflix
- UAE-IX
- ...
Agenda

- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?
What’s next?

ASPA, MANRS,...
Autonomous System Provider Authorisation

- ASPA is the next step after signing ROAs and implementing ROV
  - ASPA is a digitally signed object that binds, for a selected address family, a Set of Provider AS numbers to a Customer AS number (in terms of BGP announcements)
  - The object is signed by the holder of the Customer AS
    - The AS holder is signing who their adjacent ASes are
  - The ASPA record attests that the Customer AS has authorised the Set of Provider ASes to propagate the customer’s IPv4/IPv6 announcements onwards
Once the customer has signed their ASPA attestation:

- The neighbour AS providers (relying party) need to have access to the complete set of cryptographically valid ASPAs
- The relying party retrieves all cryptographically valid ASPAs for the customer AS
  - If none exist, then the outcome is “Unknown”
  - If the relying party’s AS is included, the outcome is “Valid”
  - If the relying party’s AS is NOT included, the outcome is “Invalid”
ASPA development

- Discussion ongoing in IETF SIDR Ops Working Group
  - Recent (last minute) change: removal of address family support
    - Which means that the relationship between two ASes must be the same for IPv4 and IPv6 (congruent)
    - Operationally this is not the case – in reality, peering policies for IPv4 are often different from those for IPv6
    - Could leave ASes without identical IPv4 and IPv6 peering policies open to the abuse ASPA was meant to help prevent
    - But much easier to implement!
  - Working documents:
ASPA development

- Router OS support is still in the early stages
  - NIST BGP-SRx and OpenBGPD support ASPA (May 2023)
  - BIRD being worked on
  - FRRouting & mainstream router vendors unknown

- Validator implementations are still in the early stages:
  - RPKI-client & StayRTR supports ASPA (May 2023)
  - Routinator support due soon
  - RPSTIR2 testing ASPA

- RIR support
  - Unknown – all RIRs will need to allow address holders to create ASPA objects
Routing Security

- Implement the recommendations in https://www.manrs.org

1. Prevent propagation of incorrect routing information
   - Filter BGP peers, in & out!
2. Prevent traffic with spoofed source addresses
   - BCP38 – Unicast Reverse Path Forwarding
3. Facilitate communication between network operators
   - NOC to NOC Communication
   - Up-to-date details in Route and AS Objects, and PeeringDB
4. Facilitate validation of routing information
   - Route Origin Authorisation using RPKI
Summary

- Deploy RPKI
  - It is in the Internet’s best interest
- With wide deployment of RPKI it becomes possible to only allow validated prefix announcements into the Internet Routing System
  - Prevents mis-originations
  - Prevents prefix hijack
  - Makes the Internet infrastructure more reliable and more stable
  - Allows the next step: AS-PATH validation
BGP Origin Validation

ISP Workshops