BGP Origin Validation

ISP Workshops

Last updated 21st May 2024

Acknowledgements

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

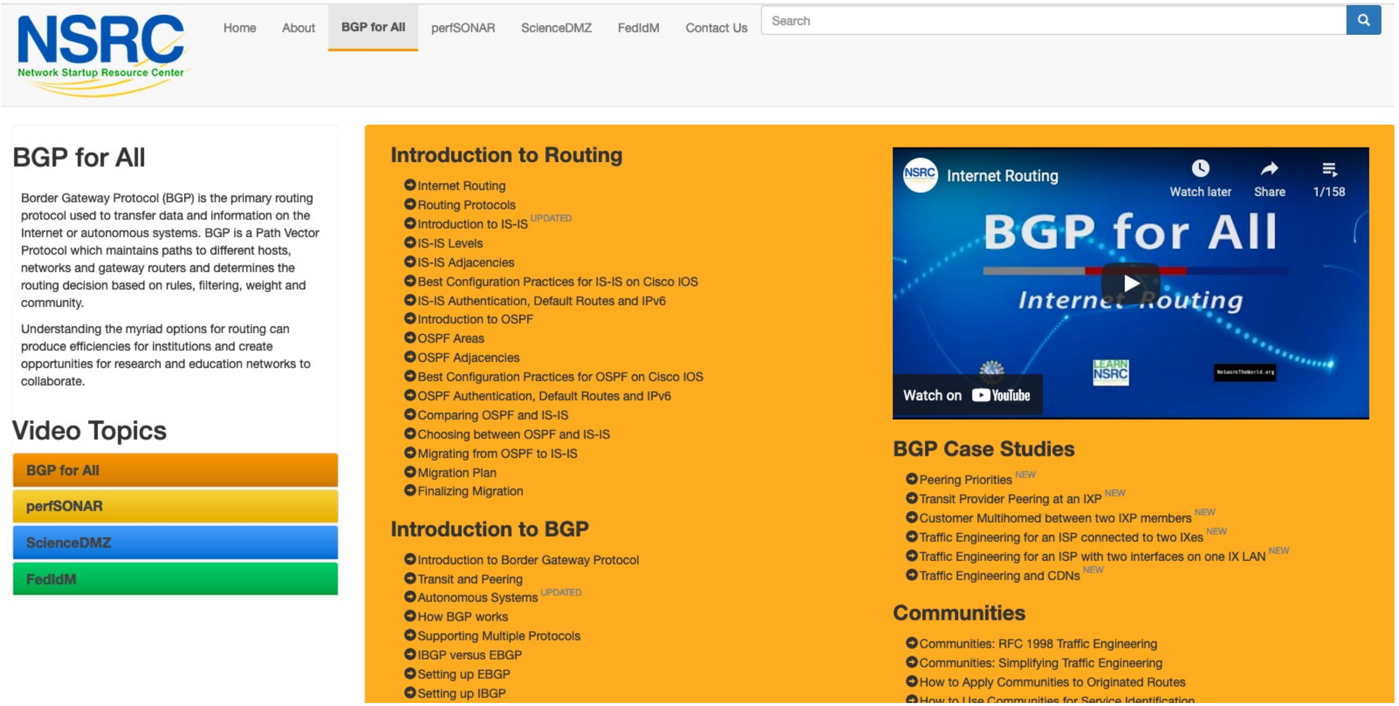
- Use of these materials is encouraged as long as the source is fully acknowledged and this notice remains in place

- 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  - And has a large amount of outdated/incorrect information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

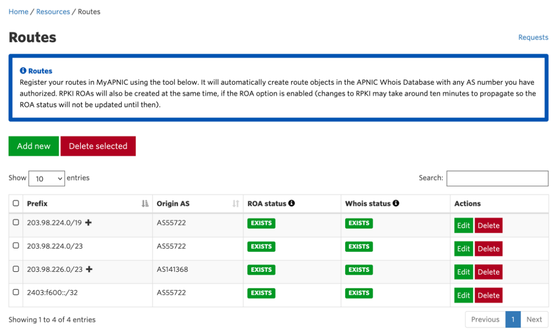
- Origin Validation
  - Using the RPKI to detect and prevent mis-originations of someone else's prefixes (RFC6483)
  - Implementation started in 2012
- AS-Path Validation
  - BGPsec has not yet begun deployment (cryptographic computation load)
  - soBGP was one early option
    - https://datatracker.ietf.org/doc/draft-white-sobgp-architecture/ (expired)
    - Not standardised or implemented
  - ASPA (Autonomous System Provider Authorisation) is one more step towards full BGPsec deployment
    - https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/

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



Home / Resources / Routes

Routes Requests

Routes
Register your routes in MyAPNIC using the tool below. It will automatically create route objects in the APNIC Whois Database with any AS number you have authorized. RPKI ROAs will also be created at the same time, if the ROA option is enabled (changes to RPKI may take around ten minutes to propagate so the ROA status will not be updated until then).

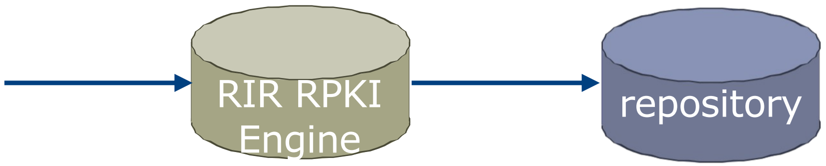
Add new | Delete selected

Show 10 entries Search:

Prefix	Origin AS	ROA status	Whois status	Actions
203.98.224.0/19 +	AS55722	EXISTS	EXISTS	Edit Delete
203.98.224.0/23	AS55722	EXISTS	EXISTS	Edit Delete
203.98.226.0/23 +	AS141368	EXISTS	EXISTS	Edit Delete
2403:f600::/32	AS55722	EXISTS	EXISTS	Edit Delete

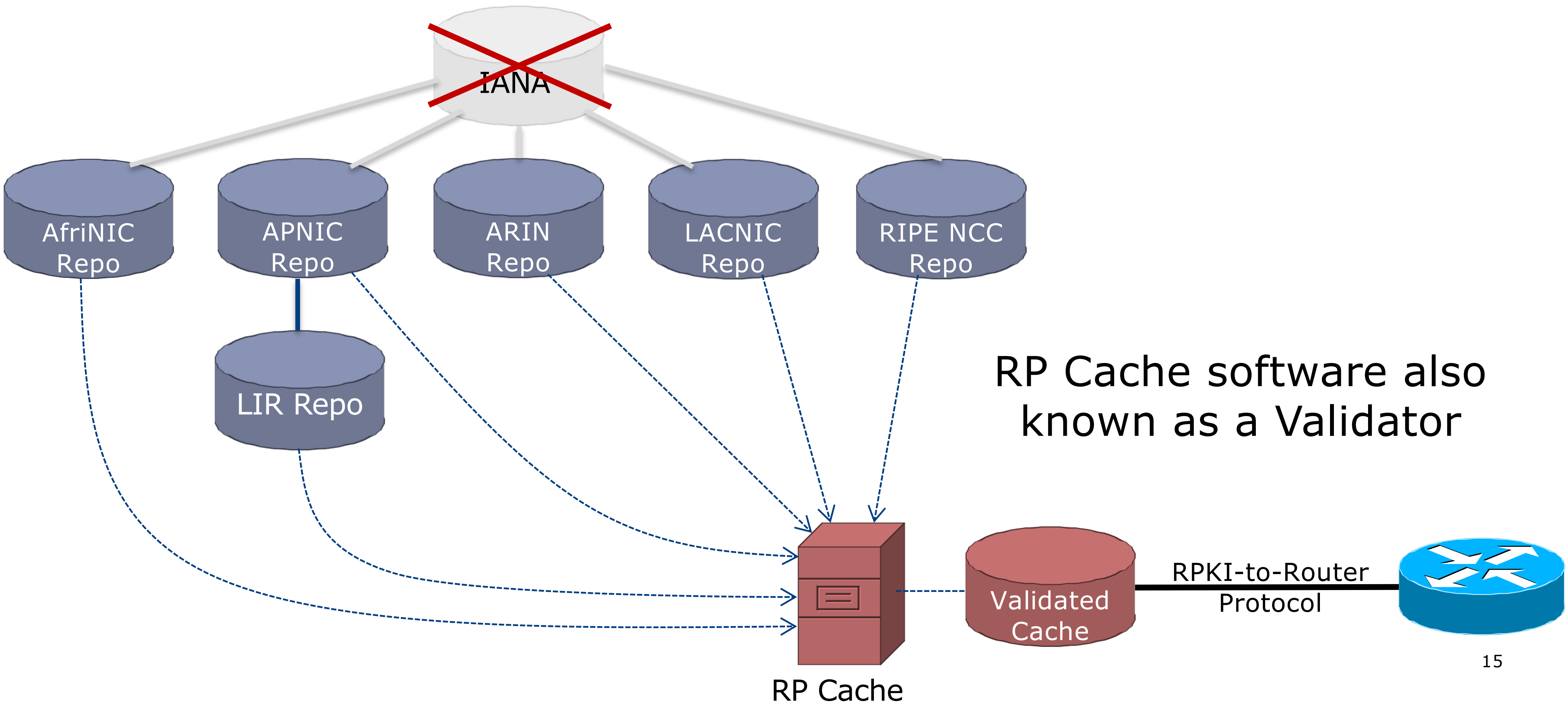
Showing 1 to 4 of 4 entries Previous 1 Next

RIR web interface



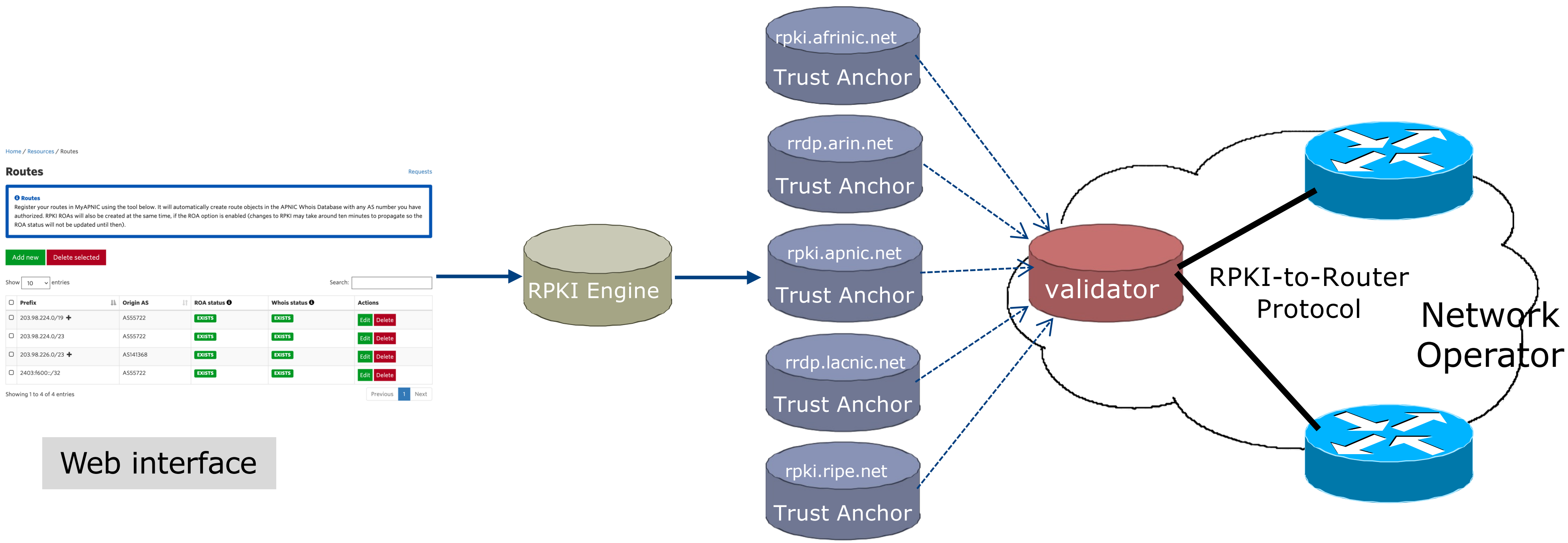
RIR public repository

Relying Party (RP)



RP Cache software also known as a Validator

RPKI Components



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:

Prefix	10.10.0.0/16
Max-Length	/18
Origin-AS	AS65534

- 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:

Prefix	Max Length	Origin AS	Comments
10.10.0.0/16	/24	65534	ROA covers /16 through to /24 – any announced subnets to /24 will be Valid if from AS65534
10.10.0.0/16	/16	65534	ROA covers only /16 – any announced subnets will be Invalid
10.10.4.0/22	/24	65534	ROA covers this /22 through to /24
10.10.32.0/22	/24	64512	Valid ROA covers /22 through to /24 announcements from AS64512

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
  - https://www.rfc-editor.org/rfc/rfc9319.html
  - 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:

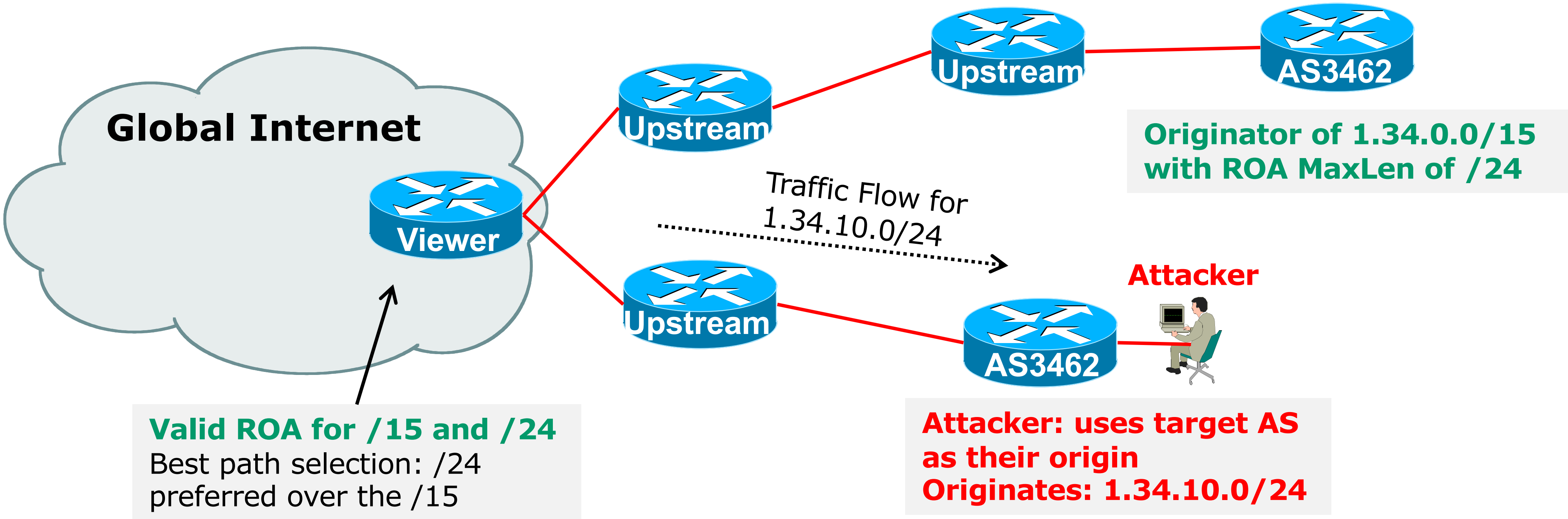
328037	2c0f:f0c8::/32	128

  - 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!

3462	1.34.0.0/15	24

  - 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

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:
    - https://www.apnic.net/wp-content/uploads/2018/02/APNIC-AR-2017.pdf
  - Example, RIPE NCC:
    - https://www.ripe.net/manage-ips-and-asns/resource-management/certification/resource-certification-rpki-for-provider-independent-end-users
  - 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:

State	Description
Valid	When authorisation is found for prefix X coming from ASN Y
Invalid	When authorisation is found for prefix X but **not** from ASN Y, or **not** allowable subnet size
Not Found	When no authorisation data is found for prefix X

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.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:

```
RPKI/RTR prefix table
Prefix                                     Prefix Length  Origin-AS
2.57.180.0                                    22 -  24            0
5.57.80.0                                     22 -  22            0
23.4.85.0                                     24 -  24            0
23.173.176.0                                  24 -  24            0
23.211.114.0                                  23 -  24            0
45.12.44.0                                    22 -  22            0
58.181.75.0                                   24 -  24            0
109.122.244.0                                 22 -  22            0
```

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://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://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://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://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
- 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

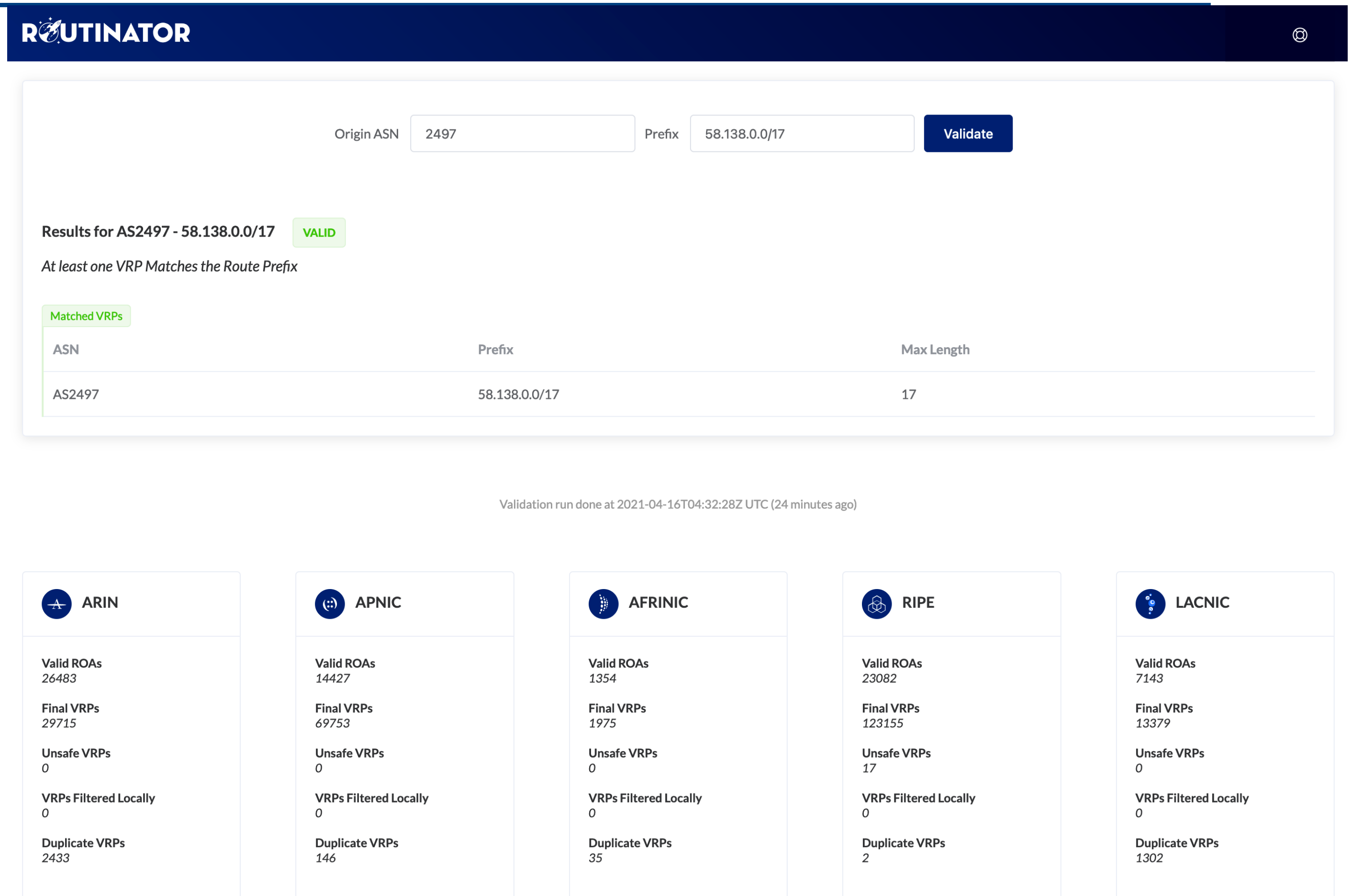
Installing a validator – Routinator

- If using Ubuntu/Debian, then simply use the package manager, as described:
  - https://github.com/NLnetLabs/routinator#quick-start-with-debian-and-ubuntu-packages
- In summary:
  - Get the NLnetLabs public key
  - Add the repo to the sources lists
  - Install routinator
  - Initialise
  - Run

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:

```
philip@fort:~$ wget https://github.com/NICMx/FORT-validator/releases/download/1.5.3/fo
rt_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: 214136 (209K) [application/octet-stream]
Saving to: ‘fort_1.5.3-1_amd64.deb’

fort_1.5.3-1_amd64.d 100%[======================>] 209.12K

2022-01-20 13:00:51 (6.93 MB/s) - ‘fort_1.5.3-1_amd64.deb’

philip@fort:~$
```

```
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:
  libjansson4
The following NEW packages will be installed:
  fort libjansson4
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, 705 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 amd64 1.5.3-1
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libja
 [28.9 kB]
Fetched 28.9 kB in 1s (30.0 kB/s)
```

```
Selecting previously unselected package libjansson4:amd64.
(Reading database ... 37466 files and directories currently installed.)
Preparing to unpack .../libjansson4_2.12-1build1_amd64.deb ...
Unpacking libjansson4:amd64 (2.12-1build1) ...
Selecting previously unselected package fort.
Preparing to unpack .../philip/fort_1.5.3-1_amd64.deb ...
Unpacking fort (1.5.3-1) ...
Setting up libjansson4:amd64 (2.12-1build1) ...
Setting up fort (1.5.3-1) ...
Adding system user `fort' (UID 116) ...
Adding new group `fort' (GID 122) ...
Adding new user `fort' (UID 116) with group `fort' ...
Not creating home directory `/var/lib/fort'.
Created symlink /etc/systemd/system/multi-user.target.wants/fort.service → /lib/system
d/system/fort.service.
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
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:

```
philip@fort:~$ sudo fort --init-tals --tal=/etc/fort/tal
Jan 20 13:16:00 DBG: HTTP GET: https://rpki.afrinic.net/tal/afrinic.tal
Jan 20 13:16:02 DBG: Done. Total bytes transferred: 496
Jan 20 13:16:02 DBG: HTTP result code: 200
Successfully fetched '/etc/fort/tal/afrinic.tal'!

Jan 20 13:16:02 DBG: HTTP GET: https://tal.apnic.net/apnic.tal
Jan 20 13:16:02 DBG: Done. Total bytes transferred: 466
Jan 20 13:16:02 DBG: HTTP result code: 200
Successfully fetched '/etc/fort/tal/apnic.tal'!

Attention: ARIN requires you to agree to their Relying Party Agreement (RPA) before
you can download and use their TAL.
Please download and read https://www.arin.net/resources/manage/rpki/rpa.pdf
If you agree to the terms, type 'yes' and hit Enter: yes
Jan 20 13:16:11 DBG: HTTP GET: https://www.arin.net/resources/manage/rpki/arin.tal
Jan 20 13:16:12 DBG: Done. Total bytes transferred: 487
Jan 20 13:16:12 DBG: HTTP result code: 200
Successfully fetched '/etc/fort/tal/arin.tal'!

Jan 20 13:16:12 DBG: HTTP GET: https://www.lacnic.net/innovaportal/file/4983/1/lacni
c.tal
Jan 20 13:16:14 DBG: Done. Total bytes transferred: 502
net/ripe-ncc.tal
82
```

```
philip@fort:~$ ll /var/lib/fort
total 8
drwxr-xr-x  2 root root 4096 Nov  9 13:33 ./
drwxr-xr-x 42 root root 4096 Jan 20 13:07 ../

philip@fort:~$ sudo chown fort:fort /var/lib/fort

philip@fort:~$ ll /var/lib/fort
total 8
drwxr-xr-x 17 fort fort 4096 Jan 20 13:40 ./
drwxr-xr-x 42 root root 4096 Jan 20 13:07 ../
```

```
Jan 20 13:33:22 fort fort[5768]: Stack trace:
Jan 20 13:33:22 fort fort[5768]:   /usr/bin/fort(print_stack_trace+0x37) [0x55e4d7e
Jan 20 13:33:22 fort fort[5768]:   /usr/bin/fort(__pr_op_err+0x98) [0x55e4d7e27fc8]
Jan 20 13:33:22 fort fort[5768]:   /usr/bin/fort(handle_flags_config+0x38b) [0x55e4
Jan 20 13:33:22 fort fort[5768]:   /usr/bin/fort(main+0x66) [0x55e4d7e232c6]
Jan 20 13:33:22 fort fort[5768]:   /lib/x86_64-linux-gnu/libc.so.6(__libc_start_mai
Jan 20 13:33:22 fort fort[5768]:   /usr/bin/fort(_start+0x2a) [0x55e4d7e233fa]
Jan 20 13:33:22 fort fort[5768]: (End of stack trace)
Jan 20 13:33:22 fort systemd[1]: fort.service: Main process exited, code=exited, st
Jan 20 13:33:22 fort systemd[1]: 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:
    - https://github.com/rpki-client/rpki-client-portable

```
philip@validator:~$ git clone --depth 1 https://github.com/rpki-client/rpki-client-portable.git
Cloning into 'rpki-client-portable'...
remote: Enumerating objects: 53, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 53 (delta 4), reused 23 (delta 1), pack-reused 0
Unpacking objects: 100% (53/53), 59.90 KiB | 2.50 MiB/s, done.
```

- 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/
  - Unpack and then run:

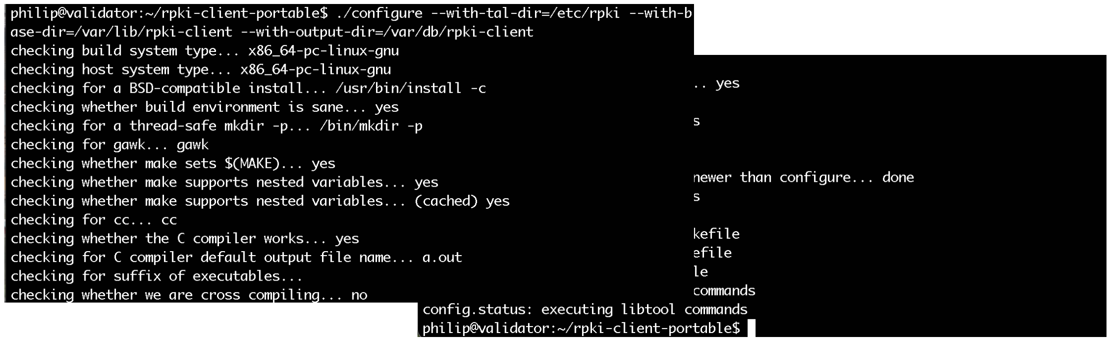
```
./configure --enable-libtls-only
make
make install
```

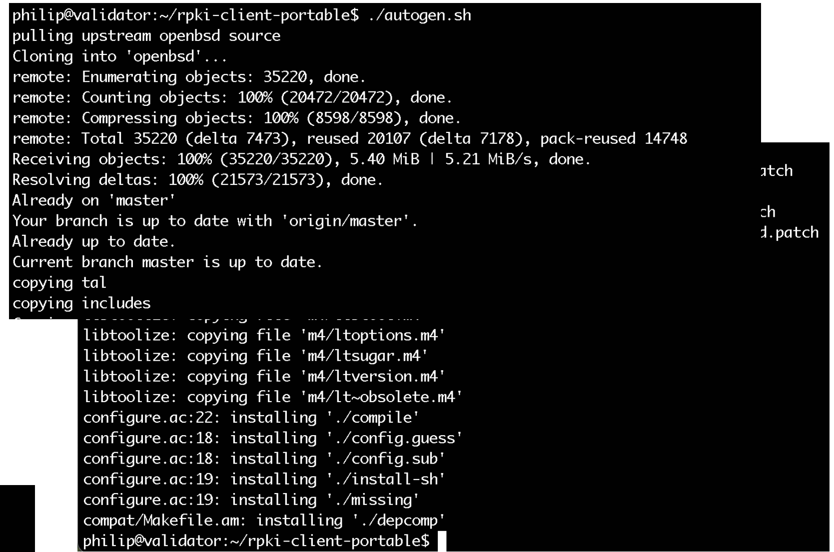
  - Which will build and install the libtls that rpki-client needs

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:

```
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:

```
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:
  - https://www.arin.net/resources/manage/rpki/arin.tal
- Now the client can be run (at the command-line, no daemon)

```
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.taaa.eu/rrdp/notification.xml: connect: Connection refused
rpki-client: https://rrdp.taaa.eu/rrdp/notification.xml: connect: Connection refused
```

- Client authors recommend running the client hourly by cron
  - See 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

```
philip@validator:~$ git clone https://github.com/bgp/stayrtr
Cloning into 'stayrtr'...
remote: Enumerating objects: 1501, done.
remote: Counting objects: 100% (1501/1501), done.
remote: Compressing objects: 100% (766/766), done.
remote: Total 1501 (delta 723), reused 1379 (delta 635), pack-reused 0
Receiving objects: 100% (1501/1501), 8.50 MiB | 7.16 MiB/s, done.
Resolving deltas: 100% (723/723), done.
```

```
philip@validator:~/stayrtr$ go build cmd/stayrtr/stayrtr.go
go: downloading github.com/prometheus/client_golang v1.11.0
go: downloading golang.org/x/crypto v0.0.0-20210921155107-089bfa567519
go: downloading github.com/sirupsen/logrus v1.8.1
go: downloading golang.org/x/sys v0.0.0-20210615035016-665e8c7367d1
go: downloading github.com/prometheus/client_model v0.2.0
go: downloading github.com/prometheus/common v0.26.0
go: downloading github.com/golang/protobuf v1.4.3
go: downloading github.com/beorn7/perks v1.0.1
go: downloading github.com/cespare/xxhash/v2 v2.1.1
go: downloading github.com/prometheus/procfs v0.6.0
go: downloading github.com/matttproud/golang_protobuf_extensions v1.0.1
go: downloading google.golang.org/protobuf v1.26.0-rc.1
```

```
philip@validator:~/stayrtr$ make build-stayrtr
mkdir -p dist/
go build -trimpath -ldflags '-X main.version=0.1-88-gf43d23e -X main.buildinfos=(2022-01-20T17:22:59+1000)' -o dist/stayrtr-0.1-88-gf43d23e-linux-x86_64 cmd/stayrtr/stayrtr.go
```

```
philip@validator:~/stayrtr$ sudo cp -p dist/stayrtr-0.1-88-gf43d23e-linux-x86_64 /usr/local/bin/stayrtr
```

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

RP Cache Deployment: Open Questions

- 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:

```
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;
      }
    }
```

```
(continued)...

    term UNKNOWN {
      from {
        protocol bgp;
        validation-database unknown;
      }
      then {
        validation-state unknown;
        next policy;
      }
    }
  }
}
```

To drop invalid prefixes, replace the `next policy` with `reject`

Configure Router to Use Cache: JunOS

3. Apply policy to eBGP session:

```
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

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!**

Check Server

```
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, Mininum 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

```
396346 BGP sovc network entries using 63415360 bytes of memory
440705 BGP sovc record entries using 14102560 bytes of memory

Network                 Maxlen  Origin-AS  Source  Neighbor
1.0.0.0/24              24      13335      0       128.223.157.83/3323
1.0.4.0/24              24      38803      0       128.223.157.83/3323
1.0.4.0/22              22      38803      0       128.223.157.83/3323
1.0.5.0/24              24      38803      0       128.223.157.83/3323
1.0.6.0/24              24      38803      0       128.223.157.83/3323
1.0.7.0/24              24      38803      0       128.223.157.83/3323
1.0.64.0/18             18      18144      0       128.223.157.83/3323
1.1.1.0/24              24      13335      0       128.223.157.83/3323
1.1.4.0/22              22      4134       0       128.223.157.83/3323
1.1.16.0/20             20      4134       0       128.223.157.83/3323
1.2.9.0/24              24      4134       0       128.223.157.83/3323
1.2.10.0/24             24      4134       0       128.223.157.83/3323
1.2.11.0/24             24      4134       0       128.223.157.83/3323
1.2.12.0/22             22      4134       0       128.223.157.83/3323
1.3.0.0/16              16      4134       0       128.223.157.83/3323
1.6.0.0/22              24      9583       0       128.223.157.83/3323
```

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

Network                   Maxlen  Origin-AS  Source  Neighbor
2001:200::/32             32      2500       0       128.223.157.83/3323
2001:200:136::/48         48      9367       0       128.223.157.83/3323
2001:200:1BA::/48         48      24047      0       128.223.157.83/3323
2001:200:900::/40         40      7660       0       128.223.157.83/3323
2001:200:E00::/40         40      4690       0       128.223.157.83/3323
2001:200:8000::/35        35      4690       0       128.223.157.83/3323
2001:200:C000::/35        35      23634      0       128.223.157.83/3323
2001:200:E000::/35        35      7660       0       128.223.157.83/3323
2001:201::/32             32      0          0       128.223.157.83/3323
2001:202::/31             31      0          0       128.223.157.83/3323
2001:204::/30             30      0          0       128.223.157.83/3323
2001:209::/32             32      0          0       128.223.157.83/3323
2001:20A::/31             31      0          0       128.223.157.83/3323
2001:20C::/30             30      0          0       128.223.157.83/3323
2001:210::/29             29      0          0       128.223.157.83/3323
2001:218::/32             32      2914       0       128.223.157.83/3323
```

BGP Table (IPv4)

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

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

Courtesy of RouteViews

BGP Table (IPv6)

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

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


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
```

RPKI BGP State: Invalid

```
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
```

RPKI BGP State: Not Found

```
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

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 95000 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 talk with a validator
    - Supporting ROV means dropping **invalid**s 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 **invalid**s by default – to allow **invalid**s to be distributed by IBGP, use the per address-family command:

```
bgp bestpath prefix-validate allow-invalid
```

Propagating validation state

- RFC8097 describes the propagation of validation state between iBGP speakers
  - Defines an opaque extended BGP community

Extended Community	Meaning
0x4300:0:0	Valid
0x4300:0:1	NotFound
0x4300:0:2	Invalid

  - 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…

```
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”:

```
policy-statement PEER-in {
    term VALID {
        from community RPKI-VALID;
        then {
            validation-state valid;
            next policy;
        }
    }
    term INVALID {
        from community RPKI-INVALID;
        then {
            validation-state invalid;
            next policy;
        }
    }
    term UNKNOWN {
        from community RPKI-UNKNOWN;
        then {
            validation-state unknown;
            next policy;
        }
    }
}
```

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):

```
     Network          Next Hop       Metric LocPrf Weight Path
V*>i 61.45.249.0/24   100.68.1.1          0     50      0 121 20 135534 i
N* i                  100.68.1.3          0    200      0 20 135534 i
V*>i 61.45.250.0/24   100.68.1.1          0     50      0 121 30 135535 i
N* i                  100.68.1.3          0    150      0 30 135535 i
V*>i 61.45.251.0/24   100.68.1.1          0     50      0 121 122 40 135536 i
N* i                  100.68.1.3          0    150      0 40 135536 i
```

  - 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

Propagating validation state: Cisco IOS

- 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/

Agenda

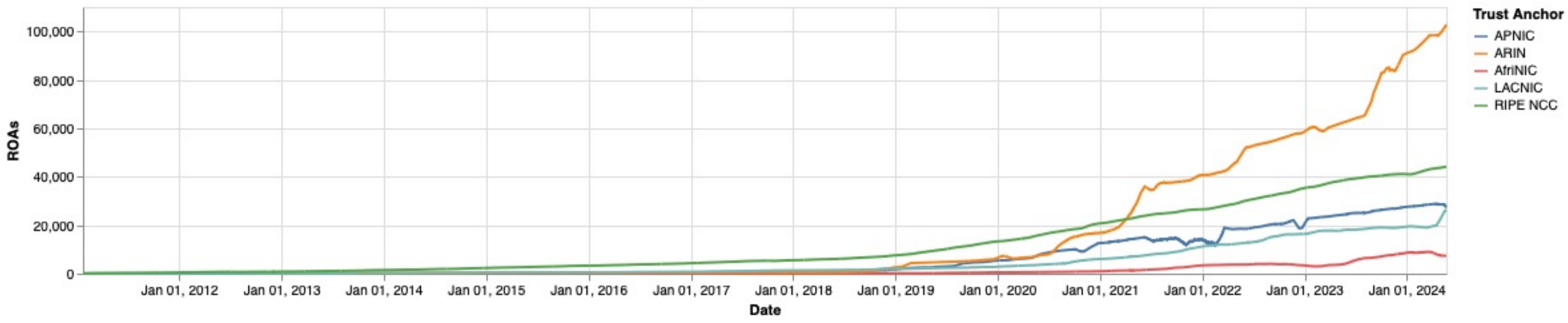
- Background – Origin Validation and RPKI
- Route Origin Authorisation
- Route Origin Validation
- Validator Caches
- Deploying RPKI
- RPKI Deployment Status
- What’s Next?

RPKI Deployment Status

RPKI Deployment Status

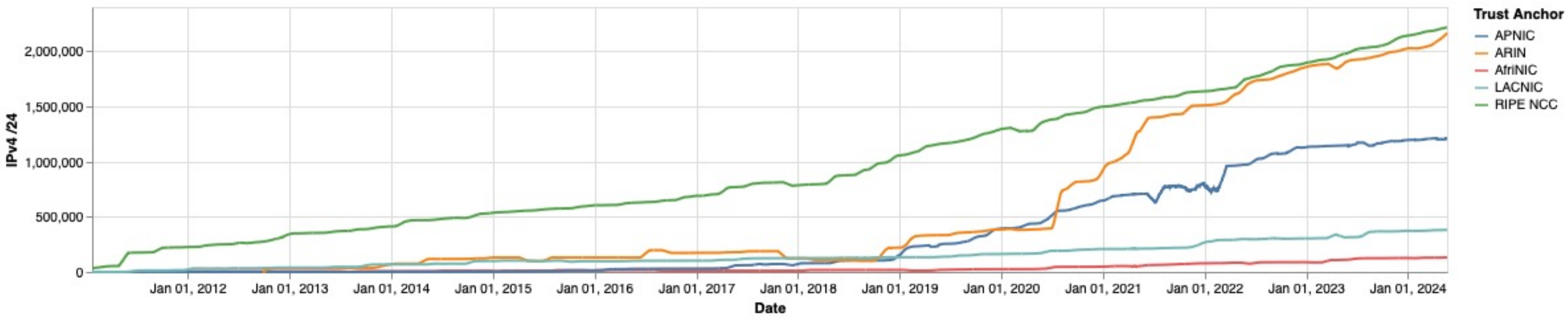
- NIST keeps track of deployment status for research purposes:
  - https://rpki-monitor.antd.nist.gov/
- IIJ Labs RPKI statistics:
  - https://ihr.iijlab.net/ihr/en-us/rov
- RIPE NCC statistics:
  - https://certification-stats.ripe.net/
- NSRC ROA status:
  - Routinator Validator running at NSRC
  - https://routinator.nsrc.org/

Number of ROAs



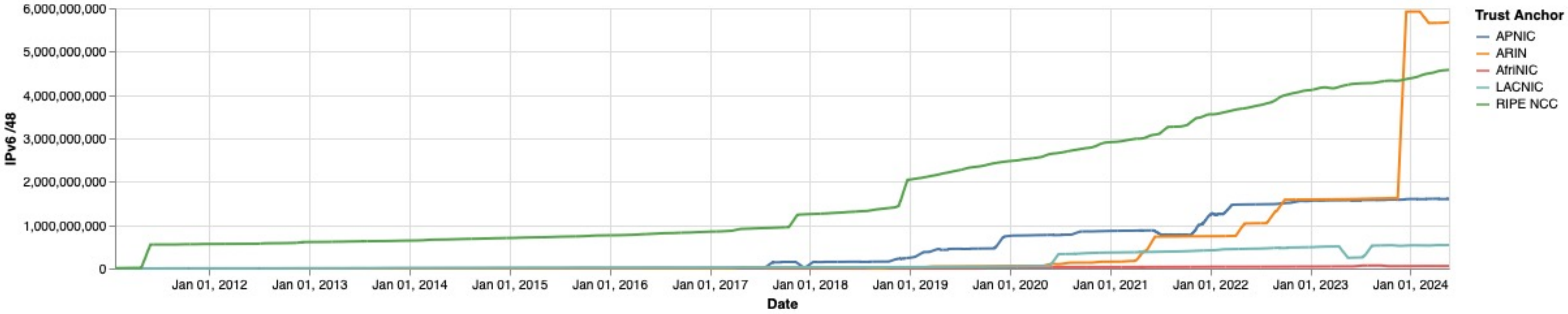
https://certification-stats.ripe.net/

IPv4 Address Space in ROAs (/24s)



https://certification-stats.ripe.net/

IPv6 Address Space in ROAs (/48s)



https://certification-stats.ripe.net/

Enter an AS, IXP, or country name

Home | Reports | Tools | Documentation | API | Contact

Route Origin Validation

IIJ Labs RPKI Statistics

3-day report ending on 21 May 2024

RPKI invalid

ROUTES | ORIGIN ASES | MAIN TRANSITS | API

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

Search

Route			Status					AS dependency
Country	Origin ASN	Prefix	RPKI	IRR	Prefix	Origin ASN	Visibility ↓	Main Transits
US	AS6939	45.12.83.0/24 autogen	Invalid	Invalid	assigned	assigned	66.8%	
SC	AS132839	156.255.216.0/24 ICIDC Limited	Invalid	Valid	assigned	assigned	65.2%	**AS13335**
ZZ	AS13335	103.21.244.0/24 Cloudflare Hong Kong, LLC 101 Townsend Street	Invalid	Invalid (more specific)	assigned	assigned	63.6%	
HK	AS135357	156.254.32.0/19 CMI IP Transit	Invalid	Invalid	assigned	assigned	63.6%	**AS13335**
US	AS13335	68.67.65.0/24 SBA EDGE COLOCATION CUSTOMERS (former GORACK) - Arelion announce	Invalid	Invalid	assigned	assigned	63.6%	
VG	AS209242	194.40.241.0/24	Invalid	Invalid	assigned	assigned	61.5%	**AS13335**
ZA	AS209242	154.16.94.0/24 AS209242	Invalid	Valid	assigned	assigned	59.9%	**AS13335**
ZZ	AS13335	2606:4700:7000::/48 101 Townsend Street, San Francisco, California 94107, US	Invalid	Invalid (more specific)	assigned	assigned	59.6%	
AU	AS36040	202.172.96.0/19 Proxy route object registered by AS38195	Invalid	Invalid	assigned	assigned	56.7%	**AS15830, AS17819, AS18390, AS38195,** AS3356 , AS6461 , AS7473
KH	AS54994	45.64.127.0/24 Proxy-registered route object	Invalid	Invalid	assigned	assigned	56.7%	**AS38623, AS9304,** AS6762 , AS7473 , AS6461

IIJ Labs RPKI Statistics

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

Enter an AS, IXP, or country name

Home | Reports | Tools | Documentation | API | Contact

Route Origin Validation

3-day report ending on 21 May 2024

RPKI invalid

ROUTES | ORIGIN ASES | MAIN TRANSITS | API

Search

ASN	Name	RPKI invalid	RPKI invalid (more specific)	Total ↓
AS18101	RELIANCE-COMMUNICATIONS-IN Reliance Communications Ltd.DAKC MUMBAI, IN	2	359	361
AS7029	WINDSTREAM, US	1	151	152
AS4804	MPX-AS Microplex PTY LTD, AU	0	115	115
AS12389	ROSTELECOM-AS PJSC Rostelecom, RU	83	7	90
AS39891	ALJAWWALSTC-AS Saudi Telecom Company JSC, SA	80	0	80
AS21491	UGANDA-TELECOM Uganda Telecom, UG	0	77	77
AS58224	TCI Iran Telecommunication Company PJS, IR	0	55	55
AS43940	MTEL-AS Drustvo za telekomunikacije "MTEL" DOO, ME	0	51	51
AS7713	TELKOMNET-AS-AP PT Telekomunikasi Indonesia, ID	1	43	44
AS45090	TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited, CN	0	36	36

Records per page: 10 | 1-10 of 810

IIJ Labs RPKI Statistics

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

Enter an AS, IXP, or country name

Home | Reports | Tools | Documentation | API | Contact

Route Origin Validation

3-day report ending on 21 May 2024

RPKI invalid

ROUTES | ORIGIN ASES | MAIN TRANSITS | API

Search

ASN	Name	RPKI invalid	RPKI invalid (more specific)	Total ↓
AS6762	SEABONE-NET TELECOM ITALIA SPARKLE S.p.A., IT	680	1484	2164
AS6453	AS6453, US	145	348	493
AS7473	SINGTEL-AS-AP Singapore Telecommunications Ltd, SG	98	298	396
AS15412	FLAG-AS FLAG TELECOM UK LIMITED, GB	18	377	395
AS3356	LEVEL3, US	160	227	387
AS6461	ZAYO-6461, US	66	271	337
AS9498	BBIL-AP BHARTI Airtel Ltd., IN	72	170	242
AS49666	TIC-GW-AS Telecommunication Infrastructure Company, IR	24	110	134
AS4755	TATACOMM-AS TATA Communications formerly VSNL is Leading ISP, IN	56	49	105
AS39386	STC-IGW-AS Saudi Telecom Company JSC, SA	87	3	90

Records per page: 10 1-10 of 573

Major Operators deploying RPKI and ROV

- Arelion

```
aut-num:        AS1299
org:            ORG-TCA32-RIPE
as-name:        TWELVE99
descr:          Arelion, f/k/a Telia Carrier
<snip>
remarks:        AS1299 is matching RPKI validation state and reject
remarks:        invalid prefixes from peers and customers.
remarks:
remarks:        Our looking-glass at https://lg.twelve99.net/ marks
remarks:        validation state for all prefixes.
remarks:
remarks:        Please review your registered ROAs to reduce number
remarks:        of invalid prefixes.
```

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
  - https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/

ASPA implementation

- 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:
    - https://datatracker.ietf.org/doc/draft-ietf-sidrops-8210bis-10
    - https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-profile
    - https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification

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

MANRS

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