Securing Internet Routing

RPKI & Route Origin Validation

ThaiNOG, 8 May 2019

Tashi Phuntsho (tashi@apnic.net)
Senior Network Janitor/Technical Trainer
Recent - Fat-finger/Hijacks/Leaks

- Google prefix leaks – **Nov 2018**
  - Google services (G-Suite, Google search and Google analytics) affected by the leak
    - Traffic dropped at AS4809 (China Telecom)
    - ~ 74 mins

---

Looking into BGP leak incident involving @google prefixes, AS37282 out of Nigeria and China Telecom.

3:40 AM - 13 Nov 2018

54 Retweets 48 Likes

---

**Breaking**: Potential hijack underway. ThousandEyes detected intermittent availability issues to Google services from some locations. Traffic to certain Google destinations appears to be routed through an ISP in Russia & black-holed at a China Telecom gateway router.
Recent - Fat-finger/Hijacks/Leaks

• Google prefix leaks (contd...)

  - How did it happen?
    - AS37282 (MainOne) leaked Google prefixes to AS4809 (CT) at IXPN, who leaked it to other transit providers like AS20485 (TransTelecom)

https://blog.thousandeyes.com/internet-vulnerability-takes-down-google/
Recent - Fat-finger/Hijacks/Leaks

- Amazon (AS16509) Route53 hijack – April 2018
  - AS10279 (eNET) originated more specifics (/24s) of Amazon Route53’s prefix (205.251.192.0/21)
    - 205.251.192.0/24 ……. 205.251.199.0/24
    - https://ip-ranges.amazonaws.com/ip-ranges.json
  - Its peers, like AS6939 (HE), shared these routes with 100s of their own peers...
  - The motive?
    - During the period, DNS servers in the hijacked range only responded to queries for myetherwallet.com
    - Responded with addresses associated with AS41995/AS48693
Recent - Fat-finger/Hijacks/Leaks

- Route53 hijack (contd...)
  - Resolvers querying any Route53 managed names, would ask the authoritative servers controlled through the BGP hijack
    - Possibly, used an automated cert issuer to get a cert for myetherwallet.com
  - Use _THEIR_ crypto to end-users to see everything (including passwords)

https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies
Recent - Fat-finger/Hijacks/Leaks

- Bharti (AS9498) originates 103.0.0.0/10 - Dec 2017
  - ~ 2 days
  - No damage done – more than 8K specific routes!

- Google brings down Internet in Japan - Aug 2017
  - ~ 24 hours
  - Google (AS15169) leaked >130K prefixes to Verizon (AS701) – in Chicago
    - Normally ~ 50 prefixes
    - ~25K of those were NTT OCN’s (AS4713) more specifics
    - which was leaked onwards to KDDI and IIJ (and accepted)

  - Everyone who received the leaked more specifics, preferred the Verizon-Google path to reach NTT OCN!
Recent - Fat-finger/Hijacks/Leaks

• Google leak (contd...)

Before leak (JP->JP)

After leak (JP->JP)

After leak (EU->EU)

Fat-finger/Hijacks/Leaks

- YouTube (AS36561) Incident - **Feb 2008**
  - ~ 2 hours
  - AS17557 (PT) announced 208.65.153.0/24 (208.65.152.0/22)
    - Propagated by AS3491 (PCCW)
Why do we keep seeing these?

• As always, there is no E-bit (evil!)
  □ A bad routing update does not identify itself as BAD
  □ All we can do is identify GOOD updates
  □ But how do we identify what is GOOD???
Why should we worry?

• Because it’s just so easy to do bad in routing!
How do we address these?

- **Filtering!**
  - Filters with your peers, upstream(s) and customers
    - Prefix filters
    - Prefix limit
    - AS-PATH filters
    - AS-PATH limit
    - RFC 8212 – BGP default reject or something similar
Current practice
Tools & Techniques

- LOA Check
  - Whois (manual)
  - Letter of Authority
  - IRR (RPSL)
Tools & Techniques

- Look up **whois**
  - verify holder of a resource

```bash
tashi@tashi -> whois -h whois.apnic.net 202.125.96.0
% [whois.apnic.net]
% Whois data copyright terms http://www.apnic.net/db/dbcopyright.html

% Information related to '202.125.96.0 - 202.125.96.255'
% Abuse contact for '202.125.96.0 - 202.125.96.255' is 'training@apnic.net'

inetnum: 202.125.96.0 - 202.125.96.255
netname: APNICTRAINING-AP
descr: Prefix for APNICTRAINING LAB DC
country: AU
admin-c: AT480-AP
tech-c: AT480-AP
status: ALLOCATED NON-PORTABLE
mnt-by: MAINT-AU-APNICTRAINING
mnt-irt: IRT-APNICTRAINING-AU
last-modified: 2016-06-17T00:17:28Z
source: APNIC

irt: IRT-APNICTRAINING-AU
address: 6 Cordelia Street
address: South Brisbane
address: QLD 4101
e-mail: training@apnic.net
abuse-mailbox: training@apnic.net
admin-c: AT480-AP
tech-c: AT480-AP
auth: # Filtered
mnt-by: MAINT-AU-APNICTRAINING
last-modified: 2013-10-31T11:01:10Z
source: APNIC
```

- role: APNIC Training
- address: 6 Cordelia Street
- address: South Brisbane
- address: QLD 4101
- country: AU
- phone: +61 7 3858 3100
- fax-no: +61 7 3858 3199
- e-mail: training@apnic.net
- admin-c: JW3997-AP
- tech-c: JW3997-AP
- nic-hdl: AT480-AP
- mnt-by: MAINT-AU-APNICTRAINING
- last-modified: 2017-08-22T04:59:14Z
- source: APNIC

% Information related to '202.125.96.0/24AS131107'

- route: 202.125.96.0/24
- descr: Prefix for APNICTRAINING LAB DC
- origin: AS131107
- mnt-by: MAINT-AU-APNICTRAINING
- country: AU
- last-modified: 2016-06-16T23:23:00Z
- source: APNIC
Tools & Techniques

- Ask for a **Letter of Authority**
  - Absolve from any liabilities

---

31/03/2018

Letter of Authorization

To whom it may concern,

APNIC Training (AS45192) runs a lab network to reproduce technical problems faced by members to help troubleshoot specific issues.

This letter serves as an authorization for APNIC Infra (AS4608) to advertise the following address blocks:

202.125.96.0/24

As a representative of APNIC Training team, that is the owner of the subnet and ASN, I hereby declare that I am authorized to sign this LOA.

Toshi Phuntscho
Training Delivery Manager

Email: toshi@apnic.net
Phone: +61 7 3858 3114

APNIC
Asia Pacific Network Information Centre
APNIC Pty Ltd
ABN: 42 081 528 070
6 Coddella Street
PO Box 3646
South Brisbane
QLD 4101 AUSTRALIA

URL www.apnic.net
Enquiries helpdesk@apnic.net
Accounts billing@apnic.net
Phone +61 7 3858 3100
Fax +61 7 3858 3199
Tools & Techniques

- Look up (or ask to enter) details in **internet routing registries (IRR)**
  - describes route origination and inter-AS routing policies
Tools & Techniques

- **IRR**
  - Helps auto generate network (prefix/as-path) filters using RPSL tools
  - Filter out route advertisements not described in the registry

```
no ip prefix-list PEER-v4IN
ip prefix-list PEER-v4IN permit 45.64.248.0/22
ip prefix-list PEER-v4IN permit 183.7.252.0/22
ip prefix-list PEER-v4IN permit 103.7.254.0/23
ip prefix-list PEER-v4IN permit 193.245.240.0/22
ip prefix-list PEER-v4IN permit 183.245.242.0/23
ip prefix-list PEER-v4IN permit 139.2.96.0/16
ip prefix-list PEER-v4IN permit 139.2.96.0/28
ip prefix-list PEER-v4IN permit 292.89.24.0/21
ip prefix-list PEER-v4IN permit 292.144.128.0/19
ip prefix-list PEER-v4IN permit 292.144.128.0/19
ip prefix-list PEER-v4IN permit 292.144.128.0/23
ip prefix-list PEER-v4IN permit 292.144.148.0/22

no ipv6 prefix-list PEER-v6IN
ipv6 prefix-list PEER-v6IN permit 2405:d000::/32
ipv6 prefix-list PEER-v6IN permit 2405:d000::7000::/36
```
Tools & Techniques

• Problem(s) with IRR
  - No single authority model
    - How do I know if a RR entry is genuine and correct?
    - How do I differentiate between a current and a lapsed entry?
  - Many RRs
    - If two RRs contain conflicting data, which one do I trust and use?
  - Incomplete data - Not all resources are registered in an IRR
    - If a route is not in a RR, is the route invalid or is the RR just missing data?
  - Scaling
    - How do I apply IRR filters to upstream(s)?
Tools & Techniques

• Automating network filters (IRR filters) - Caution

  - IRR filters only as good as the correctness of the IRR entries
    - Might require manual overrides and offline verification of resource holders
    - Good idea to use specific sources (-S in bgpq3, -s in rtconfig) when generating filters, assuming mirrors are up to date
Back to basics – identify GOOD

• Could we use a digital signature to convey the “authority to use”?
  □ Using a private key to sign the authority, and
  □ the public key to validate the authority

• The idea being:
  □ If the holder of the resource has the private key, it can sign/authorize the use of the resource
How about trust?

- How do we build a chain of trust in this framework??
  - Follow the resource allocation/delegation hierarchy

```
IANA → RIRs → NIRs/LIRs → End Holders

V

End Holders
```

- To describe the address allocation using digital certificates
RPKI Chain of Trust

Allocation Hierarchy

Trust Anchor Certificate

Certificate chain mirrors the allocation hierarchy
RIRs hold a self-signed root certificate for all the resources they have in the registry
  • they are the Trust Anchor for the system

The root certificate signs the resource certificates for end-holder allocations
  • binds the resources to the end-holders public key

Any attestations signed by the end-holder’s private key, can now be validated up the chain of trust
X.509 Certificates recap (RFC5280)

- Associates a public key with an individual or an organization

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION</td>
<td>Version of X.509</td>
</tr>
<tr>
<td>SERIAL NUMBER</td>
<td>Uniquely identifies the certificate</td>
</tr>
<tr>
<td>SIGNATURE ALGORITHM</td>
<td>Algorithms used by the CA to sign the cert</td>
</tr>
<tr>
<td>ISSUER NAME</td>
<td>Id of the CA (that issued the cert)</td>
</tr>
<tr>
<td>VALIDITY PERIOD</td>
<td>Cert validity</td>
</tr>
<tr>
<td>SUBJECT NAME</td>
<td>Entity associated with the public key</td>
</tr>
<tr>
<td>SUBJECT PUBLIC KEY</td>
<td>Owner’s public key</td>
</tr>
<tr>
<td>EXTENSIONS (ISSUER KEY ID)</td>
<td>Identify the pub key of issuer of the cert</td>
</tr>
<tr>
<td>EXTENSIONS (SUBJECT KEY ID)</td>
<td>Extra info (owner of the cert)</td>
</tr>
<tr>
<td>EXTENSIONS (CRL)</td>
<td>Extensions (CRL)</td>
</tr>
<tr>
<td>CA DIGITAL SIGNATURE</td>
<td>Certifies the binding between the pub key &amp; subject of the cert</td>
</tr>
</tbody>
</table>
RPKI profile ~ Resource Certificates

- RFC 3779 extensions – binds a list of resources (IPv4/v6,ASN) to the subject of the certificate (private key holder)
- SIA (subject information access) contains a URI that identifies the publication point of the objects signed by the subject of the cert.
Resource Certificates

• When an address holder A (*IRs) allocates resources (IP address/ASN) to B (end holders)
  
  - A issues a public-key/resource certificat that binds the allocated address with B’s public key, all signed by A’s (CA) private key
  
  - The resource certificate proves the holder of the private key (B) is the legitimate holder of the number resource!
Route Origin Authorization (ROA)

• The resource holder (B) can now sign *authorities* using its private key, which can be validated by any third party against the TA

• For routing, the address holder can *authorize* a network (ASN) to *originate* a route into the BGP routing system, and *sign* this permission with its private key (ROA)
Route Origin Authorization (ROA)

- Digitally signed object
  - list of prefixes and the nominated ASN
  - can be verified cryptographically

<table>
<thead>
<tr>
<th>Prefix</th>
<th>203.176.32.0/19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-length</td>
<td>/24</td>
</tr>
<tr>
<td>Origin ASN</td>
<td>AS17821</td>
</tr>
</tbody>
</table>

- **Multiple ROAs can exist for the same prefix**
What can RPKI do?

- Authoritatively proof:
  - Who is the legitimate owner of an address, and
  - Identify which ASNs have the permission from the holder to originate the address

- Hence, can help:
  - prevent route hijacks/mis-origination/misconfiguration
RPKI Components

- **Issuing Party** – Internet Registries (*IRs*)
  - Certificate Authority (CA) that issues resource certificates to end-holders
  - Publishes the objects (ROAs) signed by the resource certificate holders

![Diagram of RPKI Components]

- MyAPNIC GUI
- APNIC RPKI Engine
- Publication
- Repository
  - rpki.apnic.net
RPKI Components

- **Relying Party (RP)**
  - RPKI Validator tool that gathers data (ROA) from the distributed RPKI repositories
  - Validates each entry’s signature against the TA to build a “Validated cache”
RPKI Service Models

• Hosted model:
  - The RIR (APNIC) runs the CA functions on members’ behalf
    - Manage keys, repo, etc.
    - Generate certificates for resource delegations

• Delegated model:
  - Member becomes the CA (delegated by the parent CA) and operates the full RPKI system
    - JPNIC, TWNIC, CNNIC (IDNIC in progress)
Route Origin Validation (ROV)

- Global (RPKI) Repository
- RPKI Validator/ RPKI Cache server
- rsync/RRDP
- ROA
- 2406:6400::/32-48
- 17821
- RPKI-to-Router (RtR)
- AS17821
- 2406:6400::/48
- .1/1
- .2/2
- ASXXXX
- 2406:6400::/32-48
- 17821
Route Origin Validation

- Router fetches ROA information from the validated RPKI cache
  - Crypto stripped by the validator

- BGP checks each received BGP update against the ROA information and labels them
Validation States

• **Valid**
  - the prefix and AS pair found in the database.

• **Invalid**
  - prefix is found, but origin AS is wrong, OR
  - the prefix length is longer than the maximum length

• **Not Found/Unknown**
  - No valid ROA found
    - Neither valid nor invalid (perhaps not created)
### Validation States

#### ROA

<table>
<thead>
<tr>
<th>ASN</th>
<th>Prefix</th>
<th>Max Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>65420</td>
<td>10.0.0.0/16</td>
<td>18</td>
</tr>
</tbody>
</table>

#### BGP Routes

<table>
<thead>
<tr>
<th>ASN</th>
<th>Prefix</th>
<th>RPKI State</th>
</tr>
</thead>
<tbody>
<tr>
<td>65420</td>
<td>10.0.0.0/16</td>
<td>VALID</td>
</tr>
<tr>
<td>65420</td>
<td>10.0.128.0/17</td>
<td>VALID</td>
</tr>
<tr>
<td>65421</td>
<td>10.0.0.0/16</td>
<td>INVALID</td>
</tr>
<tr>
<td>65420</td>
<td>10.0.10.0/24</td>
<td>INVALID</td>
</tr>
<tr>
<td>65430</td>
<td>10.0.0.0/8</td>
<td>NOT FOUND</td>
</tr>
</tbody>
</table>
Possible actions - RPKI states

- **Do Nothing** (observe & learn)
- **Tag with BGP communities**
  - If you have downstream customers or run a route server (IXP)
    - Let them decide
  - **Ex:**
    - **Valid** (ASN:65XX1)
    - **Not Found** (ASN:65XX2)
    - **Invalid** (ASN:65XX3)
- **Modify preference values**
  - **RFC7115** *(High, Low, Lowest)*
- **Drop Invalids**
  - ~6K IPv4 routes (might want to check your top flows)
ROV – Industry trends

• **AT&T (AS7018) drops Invalids!**
  □ 11 Feb 2019

**AT&T/as7018 now drops invalid prefixes from peers**

Jay Borkenhagen [jayb at brachium.org](mailto:jayb at brachium.org)

Mon Feb 11 14:53:45 UTC 2019

- Previous message (by thread): [BGP topological vs centralized route reflector](#)
- Next message (by thread): AT&T/as7018 now drops invalid prefixes from peers
- Messages sorted by: [date] [thread] [subject] [author]

FYI:

The AT&T/as7018 network is now dropping all RPKI-invalid route announcements that we receive from our peers.

We continue to accept invalid route announcements from our customers, at least for now. We are communicating with our customers whose invalid announcements we are propagating, informing them that these routes will be accepted by fewer and fewer networks over time.

Thanks to those of you who are publishing ROAs in the RPKI. We would also like to encourage other networks to join us in taking this step to improve the quality of routing information in the Internet.

Thanks!

Jay B.
ROV – Industry trends

- **Workonline Comms (AS37271) & SEACOM (AS37100) drops Invalids!**
  - 1 and 5 April 2019 (does not use ARIN’s TAL)

---

**[apops] RPKI ROV & Dropping of Invalids - Africa**

- **To:** apops@apops.net
  - **Subject:** [apops] RPKI ROV & Dropping of Invalids - Africa
  - **From:** Mark Tinka <mark.tinka@seacom.mu>
  - **Date:** Tue, 9 Apr 2019 14:05:03 +0200

Hello all,

In November 2018 during the ZAPF (South Africa Peering Forum) meeting in Cape Town, 3 major ISP’s in Africa announced that they would enable RPKI’s ROV (Route Origin Validation) and the dropping of Invalid routes as part of an effort to clean up the BGP Internet, on the 1st April 2019.

On the 1st of April, Workonline Communications (AS37271) enabled ROV and the dropping of Invalid routes. This applies to all eBGP sessions for IPv4 and IPv6.

On the 5th of April, SEACOM (AS37100) enabled ROV and the dropping of Invalid routes. This applies to all eBGP sessions with public peers, private peers and transit providers, both for IPv4 and IPv6. eBGP sessions toward downstream customers will follow in 3 months from now.

We are still standing by for the 3rd ISP to complete their implementation, and we are certain they will communicate with the community accordingly.

Please note that for the legal reasons previously discussed on various fora, neither Workonline Communications nor SEACOM are utilizing the ARIN TAL. As a result, any routes covered only by a ROA issued under the ARIN TAL will fall back to a status of Not Found. Unfortunately, this means that ARIN members will not see any improved routing security for their prefixes on our networks until this is resolved. We will each re-evaluate this decision if and when ARIN’s policy changes. We are hopeful that this will happen sooner rather than later.

If you interconnect with either of us and may be experiencing any routing issues potentially related to this new policy, please feel free to reach out to:

- ncc@workonline.africa
- peering@seacom.mu

Workonline Communications and SEACOM hope that this move encourages the rest of the ISP community around the world to ramp up their deployment of RPKI ROV and dropping of Invalid routes, as we appreciate the work that AT&T have carried out in the same vein.

In the mean time, we are happy to answer any questions you may have about our deployments. Thanks.

Mark Tinka (SEACOM) & Ben Maddison (Workonline Communications).
Are ROAs enough?

• What if I forge the origin AS in the AS path?
  □ Would be accepted as “good” – pass origin validation!

• Which means, we need to secure the AS path as well
  □ Need AS path validation (per-prefix)

• We can use RPKI certificates for this
AS keys (per-router keys)

IANA
Cert (CA)

APNIC
Cert (CA)

APNIC Training
Cert (CA)

APNIC Training CA
202.125.96.0/24
AS45192
Public Key

APNIC Training CA
202.125.96.0/24
AS45192
Public Key

Prefix EE
202.125.96.0/24
Public Key

ROA
202.125.96.0/24
AS45192

AS Cert CA
AS45192
Public Key

I
Router EE
AS45192
rtr-00
Public Key

Encodes ASN and router ID
AS path validation - BGPsec

- AS1 router crypto signs the message to AS2
- AS2 router signs the message to AS3 and AS4, encapsulating AS1’s message

- A BGPsec speaker validates the received update by checking:
  - If there is a ROA that describes the prefix and origin AS
  - If the received AS path can be validated as a chain of signatures (for each AS in the AS path) using the AS keys
So why is AS path validation NOT happening?

- Cannot have partial adoption
  - Cannot jump across non-participating networks

- More HW resources
  - CPU - high crypto overhead to validate signatures, and
  - Memory
    - Updates in BGPsec would be per prefix
    - New attributes carrying signatures and certs/key IDs for every AS in the AS path

- No clarity on how to distribute the collection of certificates required to validate the signatures

- Given so much overhead, can it prevent more than route hijacks?
  - Route leaks?
RPKI Further Reading

- X.509 PKI Certificates
- Extensions for IP Addresses and ASNs
- Resource Public Key Infrastructure
Acknowledgement

- Geoff Huston, APNIC
- Randy Bush, IIJ Labs/Arrcus
Implementation
Create & publish your ROA

- MyApnic portal
  - Resources > RPKI

Here is a detailed guide:

Create (publish) your ROA

- Available prefixes for which you can create ROA

### BGP Route Validity

<table>
<thead>
<tr>
<th>Origin AS</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>45192</td>
<td>2001:df2:ee01::/48</td>
</tr>
<tr>
<td>45192</td>
<td>202.125.97.0/24</td>
</tr>
<tr>
<td>131107</td>
<td>2001:df2:ee00::/48</td>
</tr>
<tr>
<td>131107</td>
<td>202.125.96.0/24</td>
</tr>
<tr>
<td>135533</td>
<td>61.45.248.0/24</td>
</tr>
<tr>
<td>135540</td>
<td>61.45.248.0/24</td>
</tr>
</tbody>
</table>

Showing 1 to 6 of 6 entries

[Submit ROAs]
Create (publish) your ROA

### ROA Configuration

<table>
<thead>
<tr>
<th>Origin ASN</th>
<th>Prefix</th>
<th>Max Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>131107</td>
<td>202.128.96.0/24</td>
<td>24</td>
</tr>
<tr>
<td>131107</td>
<td>2001:df2:ee00::/48</td>
<td>48</td>
</tr>
</tbody>
</table>

Show 10 entries

Search: 131107

### Certified Resources

- 61.45.248.0/21
- 202.128.96.0/23
- 203.30.127.0/24
- 2001:0f0:0a::/48
- 2001:df2:ee00::/47
- 2406:6400::/32

Commit
Check your ROA

Validated ROAs

Validated ROAs from APNIC RPKI Root, ARIN, AfriNIC RPKI Root, LACNIC RPKI Root, RIPE NCC RPKI Root.

<table>
<thead>
<tr>
<th>ASN</th>
<th>Prefix</th>
<th>Maximum Length</th>
<th>Trust Anchor</th>
</tr>
</thead>
<tbody>
<tr>
<td>135533</td>
<td>61.45.248.0/24</td>
<td>24</td>
<td>APNIC RPKI Root</td>
</tr>
</tbody>
</table>

Showing 1 to 1 of 1 entries (filtered from 83,128 total entries)

http://nong.rand.apnic.net:8080/roas
Check your ROA

# whois -h rr.ntt.net 2001:df2:ee00::/48

route6: 2001:df2:ee00::/48
descr: RPKI ROA for 2001:df2:ee00::/48
remarks: This route object represents routing data retrieved from the RPKI
remarks: The original data can be found here: https://rpki.gin.ntt.net/r/AS131107/2001:df2:ee00::/48
remarks: This route object is the result of an automated RPKI-to-IRR conversion process.
remarks: maxLength 48
origin: AS131107
mnt-by: MAINT-JOB
changed: job@ntt.net 20180802
source: RPKI # Trust Anchor: APNIC RPKI Root
Check your ROA

# whois -h whois.bgpmon.net 2001:df2:ee00::/48

Prefix: 2001:df2:ee00::/48
Prefix description: APNICTRAINING-DC
Country code: AU
Origin AS: 131107
Origin AS Name: APNICTRAINING LAB DC
RPKI status: ROA validation successful
First seen: 2016-06-30
Last seen: 2018-01-21
Seen by #peers: 97

# whois -h whois.bgpmon.net "--roa 131107 2001:df2:ee00::/48"

------------------------
ROA Details
------------------------
Origin ASN: AS131107
Not valid Before: 2016-09-07 02:10:04
Not valid After: 2020-01-21 00:00:00 Expires in 2y190d9h34m23.200000029802s
Trust Anchor: rpki.apnic.net
Prefixes: 2001:df2:ee00::/48 (max length /48) 202.125.96.0/24 (max length /24)
Check your ROA

https://bgp.he.net/

<table>
<thead>
<tr>
<th>Announced By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin AS</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>AS131107</td>
</tr>
</tbody>
</table>
Deploy RPKI Validator

- Many options:
  - RIPE RPKI Validator
  - Dragon Research Labs RPKI Toolkit
    - [https://github.com/dragonresearch/rpki.net](https://github.com/dragonresearch/rpki.net)
  - Routinator
    - [https://github.com/NLnetLabs/routinator](https://github.com/NLnetLabs/routinator)
  - OctoRPKI & GoRTR (Cloudflare’s RPKI toolkit)
    - [https://github.com/cloudflare/cfrpki](https://github.com/cloudflare/cfrpki)
RIPE Validator

• Download RPKI Validator

```bash
# wget https://lirportal.ripe.net/certification/content/static/validator/rpki-validator-app-2.25-dist.tar.gz
```

• Installation

```bash
tar -zxvf rpki-validator-app-2.25-dist.tar.gz
cd rpki-validator-app-2.25
./rpki-validator.sh start
```

- Need to download ARIN’s TAL separately

```bash
wget https://www.arin.net/resources/rpki/arin-ripevalidator.tal
```

  - Move it to “<base-folder>/conf/tal” and restart
RIPE Validator

http://rpki-validator.apnictraining.net:8080/

Configured Trust Anchors

<table>
<thead>
<tr>
<th>Enabled</th>
<th>Trust anchor</th>
<th>Processed Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APNIC RPKI Root</td>
<td>6002 0 0</td>
</tr>
<tr>
<td></td>
<td>ARIN</td>
<td>3351 0 0</td>
</tr>
<tr>
<td></td>
<td>AfriNIC RPKI Root</td>
<td>545 0 0</td>
</tr>
<tr>
<td></td>
<td>LACNIC RPKI Root</td>
<td>6082 0 0</td>
</tr>
<tr>
<td></td>
<td>RIPE NCC RPKI Root</td>
<td>25408 0 0</td>
</tr>
</tbody>
</table>

Router Sessions

This table shows all routers connected to this RPKI Validator. Requests and responses are described in RFC 6810. For debugging, please refer to rtr.log.

<table>
<thead>
<tr>
<th>Remote Address</th>
<th>Connection Time</th>
<th>Last Request Time</th>
<th>Last Request</th>
<th>Last Reply</th>
</tr>
</thead>
</table>
Dragon Research - Validator

• Installation on Ubuntu 16.04 Xenial

https://github.com/dragonresearch/rpki.net/blob/master/doc/quickstart/xenial-rp.md

• Installation

  □ Add the GPG public key

  # wget -q -O /etc/apt/trusted.gpg.d/rpki.gpg https://download.rpki.net/APTng/apt-gpg-key.gpg

  □ Add the repo to the APT source list

  # wget -q -O /etc/apt/sources.list.d/rpki.list https://download.rpki.net/APTng/rpki.xenial.list
  
  -q: quite (wget output)
  -O: output to <file>

  # apt update  # apt install rpki-rp
Dragon Research - Validator

http://rpki-dragonresearch.apnictraining.net/rcynic/

rcynic summary 2017-01-03T01:07:37Z

<table>
<thead>
<tr>
<th></th>
<th>Tainted by stale CRL</th>
<th>Object accepted</th>
<th>Manifest interval overruns certificate</th>
<th>certificate has expired</th>
<th>Tainted by stale manifest</th>
<th>Policy Qualifier CPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>None .cer</td>
<td>28</td>
<td>5041</td>
<td></td>
<td></td>
<td>28</td>
<td>638</td>
</tr>
<tr>
<td>None .crl</td>
<td>3</td>
<td>5048</td>
<td></td>
<td></td>
<td>1</td>
<td>634</td>
</tr>
<tr>
<td>None .gbr</td>
<td>5048</td>
<td>1</td>
<td></td>
<td>1</td>
<td>621</td>
<td></td>
</tr>
<tr>
<td>None .mft</td>
<td>5023</td>
<td>23803</td>
<td>1</td>
<td>1</td>
<td>28</td>
<td>2293</td>
</tr>
<tr>
<td>None .roa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>28</td>
<td>23803</td>
<td>1</td>
<td>1</td>
<td>28</td>
<td>2293</td>
</tr>
</tbody>
</table>

Overview for repository rpki.apnic.net

<table>
<thead>
<tr>
<th></th>
<th>Tainted by stale CRL</th>
<th>Object accepted</th>
<th>Manifest interval overruns certificate</th>
<th>certificate has expired</th>
<th>Tainted by stale manifest</th>
<th>Policy Qualifier CPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>None .cer</td>
<td>752</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None .crl</td>
<td>748</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None .mft</td>
<td>748</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None .roa</td>
<td>492</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2740</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current total object counts (distinct URIs)

<table>
<thead>
<tr>
<th>Repository</th>
<th>.cer</th>
<th>.crl</th>
<th>.gbr</th>
<th>.mft</th>
<th>.roa</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca.rg.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ca0.rpki.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>localcert.ripe.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>repository.lacnic.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpki-pilot.lab.diag.de</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpki-repository.nic.ad.jp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpki.afilnet.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpki.apnic.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpki.ripe.net</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Configuration (IOS)

- Establishing session with the validator

```bash
router bgp 131107
  bgp rpki server tcp <validator-IP> port <323/8282/3323> refresh 120
```

- Note:
  - Cisco IOS by default does not include invalid routes for best path selection!
  - If you don’t want to drop invalids, we need explicitly tell BGP (under respective address families)

```bash
bgp bestpath prefix-validate allow-invalid
```
Configuration (IOS)

- Policies based on validation:

```conf
route-map ROUTE-VALIDATION permit 10
  match rpki valid
  set local-preference 110
!
route-map ROUTE-VALIDATION permit 20
  match rpki not-found
  set local-preference 100
!
route-map ROUTE-VALIDATION permit 10
  match rpki invalid
  set local-preference 90
!```
Configuration (IOS)

• Apply the route-map to inbound updates

```
router bgp 131107
!---output omitted------!
 address-family ipv4
  bgp bestpath prefix-validate allow-invalid
  neighbor X.X.X.169 activate
  neighbor X.X.X.169 route-map ROUTE-VALIDATION in
  exit-address-family
!
 address-family ipv6
  bgp bestpath prefix-validate allow-invalid
  neighbor X6:X6:X6:X6::151 activate
  neighbor X6:X6:X6:X6::151 route-map ROUTE-VALIDATION in
  exit-address-family
!
```
Configuration (JunOS)

- Establishing session with the validator

```xml
routing-options {
    autonomous-system 131107;
    validation {
        group rpki-validator {
            session <validator-IP> {
                refresh-time 120;
                port <323/3323/8282>;
                local-address X.X.X.253;
            }
        }
    }
}
```
Configuration (JunOS)

- Define policies based on the validation states

```conf
policy-options {
    policy-statement ROUTE-VALIDATION {
        term valid {
            from {
                protocol bgp;
                validation-database valid;
            }
            then {
                local-preference 110;
                validation-state valid;
                accept;
            }
        }
        term invalid {
            from {
                protocol bgp;
                validation-database invalid;
            }
            then {
                local-preference 90;
                validation-state invalid;
                accept;
            }
        }
        term unknown {
            from {
                protocol bgp;
                validation-database unknown;
            }
            then {
                local-preference 100;
                validation-state unknown;
                accept;
            }
        }
    }
}
```
Router Configuration (JunOS)

• Apply the policy to inbound updates

```plaintext
protocols {
  bgp {
    group external-peers {
      #output-ommitted
      neighbor X.X.X.1 {
        import ROUTE-VALIDATION;
        family inet {
          unicast;
        }
      }
    }
    group external-peers-v6 {
      #output-ommitted
      neighbor X6:X6:X6:X6::1 {
        import ROUTE-VALIDATION;
        family inet6 {
          unicast;
        }
      }
    }
  }
}
```
RPKI Verification (IOS)

- IOS has only

```
#sh bgp ipv6 unicast rpki ?
  servers  Display RPKI cache server information
  table    Display RPKI table entries

#sh bgp ipv4 unicast rpki ?
  servers  Display RPKI cache server information
  table    Display RPKI table entries
```
RPKI Verification (IOS)

- Check the RTR session

```
#sh bgp ipv4 unicast rpki servers
BGP SOVC neighbor is X.X.X.47/323 connected to port 323
Flags 64, Refresh time is 120, Serial number is 1516477445, Session ID is 8871
InQ has 0 messages, OutQ has 0 messages, formatted msg 7826
Session IO flags 3, Session flags 4008
  Neighbor Statistics:
  Prefixes 45661
  Connection attempts: 1
  Connection failures: 0
  Errors sent: 0
  Errors received: 0

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: X.X.X.225, Local port: 29831
Foreign host: X.X.X.47, Foreign port: 323
```
# RPKI Verification (IOS)

- Check the RPKI cache

```bash
#sh bgp ipv4 unicast rpki table
37868 BGP sovc network entries using 6058880 bytes of memory
39655 BGP sovc record entries using 1268960 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>1.9.0.0/16</td>
<td>24</td>
<td>4788</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
<tr>
<td>1.9.12.0/24</td>
<td>24</td>
<td>65037</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
<tr>
<td>1.9.21.0/24</td>
<td>24</td>
<td>24514</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
<tr>
<td>1.9.23.0/24</td>
<td>24</td>
<td>65120</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
</tbody>
</table>

#sh bgp ipv6 unicast rpki table
5309 BGP sovc network entries using 976856 bytes of memory
6006 BGP sovc record entries using 192192 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>202.125.96.47/323</td>
</tr>
<tr>
<td>2001:200:900::/40</td>
<td>40</td>
<td>7660</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
<tr>
<td>2001:200:8000::/35</td>
<td>35</td>
<td>4690</td>
<td>0</td>
<td>202.125.96.47/323</td>
</tr>
</tbody>
</table>
```
Check routes (IOS)

#sh bgp ipv4 unicast 202.144.128.0/19
BGP routing table entry for 202.144.128.0/19, version 3814371
Paths: (1 available, best #1, table default)
Advertised to update-groups:
  2
  Refresh Epoch 15
  4826 17660
  49.255.232.169 from 49.255.232.169 (114.31.194.12)
    Origin IGP, metric 0, localpref 110, valid, external, best
    Community: 4826:5101 4826:6570 4826:51011 24115:17660
    path 7F50C7CD98C8 RPKI State valid
    rx pathid: 0, tx pathid: 0x0

#sh bgp ipv6 unicast 2402:7800::/32
BGP routing table entry for 2402:7800::/32, version 1157916
Paths: (1 available, best #1, table default)
Advertised to update-groups:
  2
  Refresh Epoch 15
  4826
    2402:7800:10:2::151 from 2402:7800:10:2::151 (114.31.194.12)
      Origin IGP, metric 0, localpref 100, valid, external, best
      path 7F50B266CBD8 RPKI State not found
      rx pathid: 0, tx pathid: 0x0
## RPKI Verification (JunOS)

- Check the RPKI cache

```
>show validation session
Session           State Flaps  Uptime #IPv4/IPv6 records
X.X.X.46          Up             75 09:20:59  40894/6747

>show validation session 202.125.96.46
Session           State Flaps  Uptime #IPv4/IPv6 records
X.X.X.46          Up             75 09:21:18  40894/6747
```
RPKI Verification (JunOS)

- Check the RPKI cache

```plaintext
>show validation database
RV database for instance master

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Origin-AS</th>
<th>Session</th>
<th>State</th>
<th>Mismatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9.0.0/16-24</td>
<td>4788</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>1.9.12.0/24-24</td>
<td>65037</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>1.9.21.0/24-24</td>
<td>24514</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>1.9.23.0/24-24</td>
<td>65120</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>2001:200::/32-32</td>
<td>2500</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>2001:200:900::/40-40</td>
<td>7660</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>2001:200:c000::/35-35</td>
<td>23634</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>2001:200:e000::/35-35</td>
<td>7660</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
</tbody>
</table>
```

Would have been nice if they had per AF!
**RPKI Verification (JunOS)**

- Can filter per origin ASN

```bash
>show validation database origin-autonomous-system 45192
RV database for instance master

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Origin-AS</th>
<th>Session</th>
<th>State</th>
<th>Mismatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>202.125.97.0/24-24</td>
<td>45192</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>203.176.189.0/24-24</td>
<td>45192</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
<tr>
<td>2001:df2:ee01::/48-48</td>
<td>45192</td>
<td>202.125.96.46</td>
<td>valid</td>
<td></td>
</tr>
</tbody>
</table>

IPv4 records: 2
IPv6 records: 1
```

*IOS should have something similar!*
Check routes (JunOS)

>show route protocol bgp 202.144.128.0

inet.0: 693024 destinations, 693024 routes (693022 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both

202.144.128.0/20 *[BGP/170] 1w4d 21:03:04, MED 0, localpref 110, from 202.125.96.254
  AS path: 4826 17660 I, validation-state: valid
  >to 202.125.96.225 via ge-1/1/0.0

>show route protocol bgp 2001:201::/32

inet6.0: 93909 destinations, 93910 routes (93909 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:201::/32 *[BGP/170] 21:18:14, MED 0, localpref 100, from 2001:df2:ee00::1
  AS path: 65332 I, validation-state: unknown
  >to fe80::dab1:90ff:fedc:fd07 via ge-1/1/0.0
Propagating RPKI states to iBGP peers

- To avoid every BGP speaker having an RTR session, and
- All BGP speakers have consistent information

  - Relies on extended BGP communities (RFC8097)

```
0  | 1  | 2  | 3
0 1234567890123456789012345678901
+----------------+----------------+----------------+----------------+
| 0x43 | 0x00 | Reserved |
+----------------+----------------+----------------+
| Reserved | [validation state] |
+----------------+----------------+
```

- Sender (one that has RTR session) attaches the extended community to Updates, and receiver derives the validation states from it
- Must be enabled on both sender and receiver!
Propagating RPKI states (IOS)

- **Sender (one with RTR session)**

```conf
router bgp 131107
  bgp rpki server tcp <validator-IP> port <323/8282/3323> refresh 120
  !--- output omitted ----!
  address-family ipv4
  neighbor X.X.X.X activate
  neighbor X.X.X.X send-community both
  neighbor X.X.X.X announce rpki state
exit-address-family
!
address-family ipv6
neighbor X6:X6:X6::X6 activate
neighbor X6:X6:X6::X6 send-community both
neighbor X6:X6:X6::X6 announce rpki state
exit-address-family
!
```
Propagating RPKI states (IOS)

• Receiver (iBGP peer)

```
router bgp 131107
!---output omitted------!
 address-family ipv4
   neighbor Y.Y.Y.Y activate
   neighbor Y.Y.Y.Y send-community both
   neighbor Y.Y.Y.Y announce rpki state
 exit-address-family

 address-family ipv6
   neighbor Y6:Y6:Y6:Y6::Y6 activate
   neighbor Y6:Y6:Y6:Y6::Y6 send-community both
   neighbor Y6:Y6:Y6:Y6::Y6 announce rpki state
 exit-address-family
```

- If `announce rpki state` is not configured for the neighbor, all prefixes received from the iBGP neighbor will be marked VALID!
Propagating RPKI states (JunOS)

- **Sender (one with RTR session)**

```plaintext
class ROUTE-VALIDATION {
  term valid {
    from {
      protocol bgp;
      validation-database valid;
    }
    then {
      local-preference 110;
      validation-state valid;
      community add origin-validation-state-valid;
      accept;
    }
  }
  term invalid {
    from {
      protocol bgp;
      validation-database invalid;
    }
    then {
      local-preference 90;
      validation-state invalid;
      community add origin-validation-state-invalid;
      accept;
    }
  }
  term unknown {
    from {
      protocol bgp;
      validation-database unknown;
    }
    then {
      local-preference 100;
      validation-state unknown;
      community add origin-validation-state-unknown;
      accept;
    }
  }
}
```
Propagating RPKI states (JunOS)

• Receiver (iBGP peer)

```
policy-statement ROUTE-VALIDATION-1 {
  term valid {
    from community origin-validation-state-valid;
    then validation-state valid;
  }
  term invalid {
    from community origin-validation-state-invalid;
    then validation-state invalid;
  }
  term unknown {
    from community origin-validation-state-unknown;
    then validation-state unknown;
  }
}
```
Propagating RPKI states – potential issues

- IOS as BR, propagating states to JunOS iBGP peers
  
  Hack:
  - Either act on the states at the border, or
  - Match and tag them with custom communities before propagating
Configuration - Reference Link

- **Cisco**

- **Juniper**

- **RIPE:**
Operational Caveats

• When RTR session goes down, the validation state changes to **Not Found** for all routes after a while
  - **Invalid** => **Not Found**
  - we need at least two RTR sessions and/or need careful filtering policies

• During a router reload, do we receive ROAs first or BGP updates first?
  - If BGP update is faster than ROA, will propagate even invalid routes to its iBGP peers
Useful tools

- RIPEstat – prefix/ASN
  - https://stat.ripe.net/data/rpki-validation/data.json?resource=45192&prefix=202.125.96.0/24
Any questions?