

RPKI Asgroups (or, what next steps to sunset IRR/RPSL – if any?)

<https://www.ietf.org/id/draft-spaghetti-sidrops-rpki-asgroup-00.html>

Job Snijders
job@fastly.com

- Co-author of [bgpq4](#) and [irrtree](#)
- Initiator of the [IRRd](#) v4 project
- Creator of [irrexplorer](#) v1
- Co-author of [rpki-client](#)

Today - IRR/RPSL and RPKI co-exist

- **Operators use both information systems**
 - By necessity (tradition, customer expectations, vendor requirements)
 - An unavoidable side-effect of *incremental* deployment of the RPKI
- **What's what?**
 - IRR is plain-text, no authenticity, no integrity, vulnerable to replay
 - The RPKI is robust future-proof resource, a framework to build on
- **Co-existence being unavoidable, we should focus on**
 - *Incentivizing* migration (rather than mandating it through flag days)

So... what features or aspects or data need to be migrated?

Some analogies between IRR/RPSL and RPKI

None of these are isomorphic! (which is good, I suppose)

mntner: object	→ RPKI X.509 Certification Authority
route:/route6: object	→ RPKI ROA structure
person: object	→ RPKI GhostBuster record
aut-num export-via:	→ ASPA (<i>*squints eyes*</i>)

Examples where no analogies exist at all:

- IRR/RPSL database completeness cannot be verified, while RPKI has Manifests
- RPKI validation paths are short-lived, expiry is visible to RP; while IRR/RPSL objects exist until cows come home

Example of successful IRR/RPSL → RPKI migration

As explained in 2018:

http://iepg.org/2018-11-04-ietf103/IETF103_Snijders_Routing_security_roadmap-v2.pdf

IRRD version 4 suppresses RPKI-invalid **route: objects**

<https://irrd.readthedocs.io/en/stable/admins/rpki/>

NTT's **rr.ntt.net** already supports this, RADB's **whois.radb.net** to follow in 2023

RIPE NCC's **RIPE - NONAUTH** database deletes RPKI-invalid **route: objects** after 14 days

<https://www.ripe.net/publications/docs/ripe-731>

Advantages for RPKI certified resource holders:

- Prevention of creation of new conflicting IRR objects
- Cleans up stale / old conflicting data



Source: <http://clipart-library.com/clipart/kiMKoBRrT.htm>

Example of ongoing pain in IRR ecosystem

Many operators (for better or worse) use **as-set**: objects in generating BGP filters

Perpetrators create same-named objects in different IRR databases to *impersonate* the victim, recent example: AS-AMAZON. In effect a Denial-of-Service attack.

Example of pain in IRR ecosystem: AS-AMAZON

```
good$ whois -h whois.radb.net AS-AMAZON
as-set:      AS-AMAZON
descr:      Amazon ASNs
members:    AS-AMAZON-NA, AS-AMAZON-AP, AS-AMAZON-EU
admin-c:    AC6-ORG-ARIN
tech-c:     AC6-ORG-ARIN
notify:     noc@amazon.com
mnt-by:     MAINT-AS16509
changed:    noc@amazon.com 20151027 #17:32:13Z
source:     RADB
```

Example of pain in IRR ecosystem: AS-AMAZON

```
bad$ whois -h whois.ripe.net AS-AMAZON
as-set:          AS-AMAZON
tech-c:          KR4968-RIPE
admin-c:         KR4968-RIPE
mnt-by:          KATERINA-MNT
created:         2022-10-23T19:05:59Z
last-modified:   2022-10-23T19:05:59Z
source:          RIPE
```

Sources:

<https://www.ripe.net/ripe/mail/archives/db-wg/2022-November/007693.html>

<https://www.ripe.net/ripe/mail/archives/db-wg/2022-November/007649.html>

Quick hack in IRR, mandate use of namespaces:

BAD: AS-SNIJDERS

BETTER: AS15562:AS-SNIJDERS

RIPE (proposal NWI-19):

<https://www.ripe.net/ripe/mail/archives/db-wg/2022-November/007646.html>

<https://www.ripe.net/ripe/mail/archives/db-wg/2022-November/007678.html>

<https://www.ripe.net/ripe/mail/archives/db-wg/2022-November/007680.html>

APNIC (PROP-151 – Restricting non-hierarchical as-set)

<https://www.apnic.net/community/policy/proposals/prop-151/>

https://2023.apricot.net/assets/files/APPS314/as-set_1677635099.pdf

RADB: announced last week they'll require hierarchical naming in 2023 (date TBD)

LACNIC: since inception only supports as-sets following hierarchical naming

AS-SETs recursively reference by value & by name

```
$ whois -h whois.ripe.net AS15562:AS-SNIJDERS
```

```
...
```

```
as-set:          AS15562:AS-SNIJDERS
```

```
descr:          Downstream of AS 15562 and beyond
```

```
members:        AS15562      # Me
```

```
members:        AS57436     # Samer
```

```
members:        AS12654     # RIPE RIS
```

```
members:        AS-KING     # Thomas King
```

```
members:        AS-NETHER  # Jared
```

```
...
```

How does RPKI fit in all this? Strawman: AS Groups

```
RpkiSignedGrouping ::= SEQUENCE {  
  version [0]      INTEGER DEFAULT 0,  
  asID             ASID, -- the 'namespace', must be contained in RFC 3779 extension  
  label           GroupingLabel (SIZE(1..100)),  
  referenceable   BOOLEAN DEFAULT TRUE,  
  members         SEQUENCE (SIZE(0..MAX)) OF ASIdOrGroupingPointer }
```

```
ASIdOrGroupingPointer ::= CHOICE {  
  id              ASID,  
  pointer         GroupingPointer }
```

```
GroupingPointer ::= SEQUENCE {  
  asID           ASID,  
  label         GroupingLabel (SIZE(1..100)) }
```

```
ASID ::= INTEGER (1..4294967295)
```

```
GroupingLabel ::= IA5String (FROM("A".."Z" | "0".."9" | ":" | "_" | "-"))
```

ASGroups recursively reference by value & by name

```
...
asgroup: AS15562:AS-SNIJDERS # signer + label
members: AS15562 # ASID INTEGER
members: AS57436 # ASID INTEGER
members: AS12654 # ASID INTEGER
members: AS31451:AS-KING # GroupingLabel
members: AS267:AS-NETHER # GroupingLabel
...
```

What are we solving? Open questions

RPKI can be used to settle naming collisions, authorization... But what is the true purpose of IRR *as-sets*?:

“We use them all the time!”

But why?

“Security!”

Are you sure the value isn't some misunderstood second-order effect?

“We might not know...”

Are RFC9234 + ASPA a full successor both in spirit and effect?

ASGroups supports complicated machinery to securely opt-out of someone else's ASGroup, is that useful?

Should Asgroup profile have a 'purpose' enum? (“customers”, “peers”, “geographic grouping”, “other”...)

Is anyone else working on gap analysis comparing IRR and RPKI?

Open mic discussion