

ENGRIT

(Extensible Next Generation Routing Information Toolkit)

RIPE Routing WG, May 26 2016

Stavros Konstantaras

~~ENGRIT~~

mantaBGP

Something more cool?

– e-mail stavros@nlnetlabs.nl

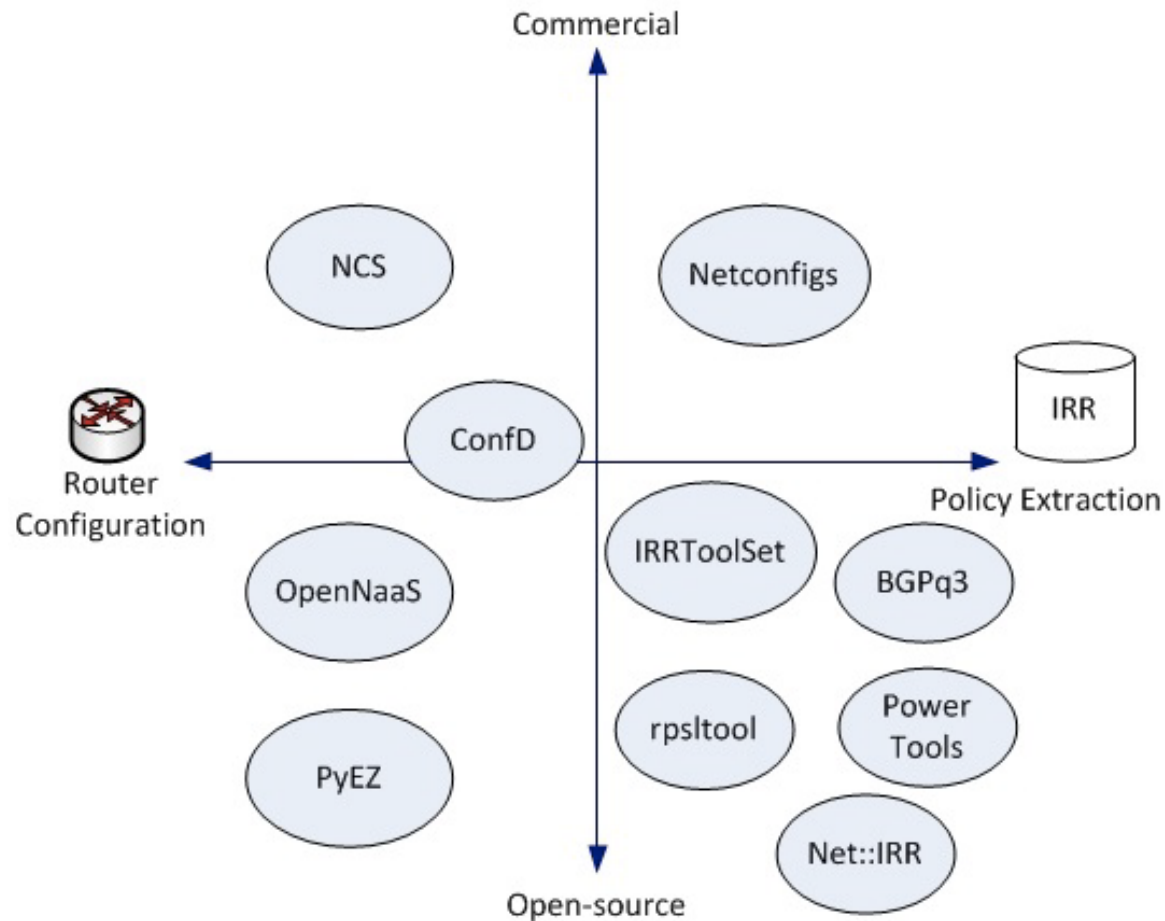
Why?

- Big Internet
 - Deaggregation (~600.000 routable prefixes)
 - IPv4 depletion
 - Complex filtering
- Misconfiguration mistakes
 - February 2008 (Pakistan Telecom): Youtube
 - April 2010 (Chinese Telecom): 15% of internet
 - November 2015 (NLnet Labs): No IPv6 traffic

Motivation

- The mission: World peace
 - Improve security and stability of the internet
- The goal: Achieve end to end configuration
 - Automate and minimize the mistakes
 - Bridge the gap between IRR and router

Related tools

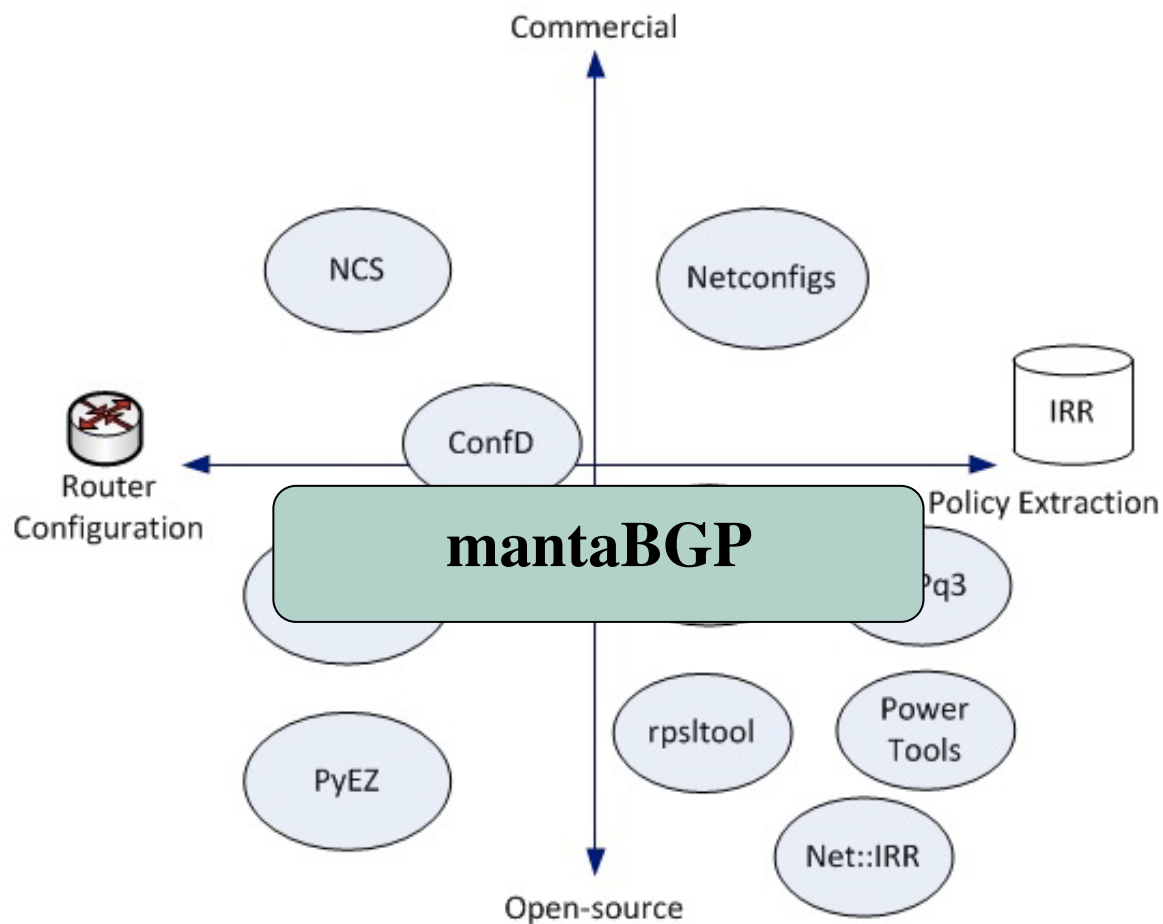


Requirements

Features	Functionalities
Flexible and extendable	Query IRR DB
32-bit ASN support	Extract peering relations
IPv6 support	Resolve peering filters
Security (via RPKI)	Use local file for extra information & tuning
Vendor independent	Push configuration to router

- Open source (BSD License)

Our position



Choices made (1/2)

- Python as a programming language
 - Easy, flexible, familiar to many network engineers
- RIPE DB as an input source of public data
 - RPSL objects retrieved by RESTful API
 - Mirrors other DBs as well

Choices made (2/2)

- NETCONF as a (universal) output
 - Supported by Juniper, Cisco, Brocade, Huawei...
- YAML based local file
 - Easy to read, understand and modify
 - Router credentials, additional routing information, fine tuning, etc.

Current status

- Finalize the input (RPSL) library:
 - A policy-parser and filter resolver
 - Will be a different branch and can be used autonomously
- Design overall tool architecture
 - Define the components and their role

Policy parser overview

- A library that converts and resolves policies into XML format
 - Uses python and REGEX
 - NOT a new peval
 - Currently supporting aut-num, route(6), AS-SETS, RS-SETS (RFC 2650)
- Try it:
 - <https://github.com/stkonst/PolicyParser>

Parser input/output

- Input: XML output of RIPE DB
 - curl
<http://rest.db.ripe.net/search.xml?query-string=as199664&type-filter=route6&inverse-attribute=origin>
- Output: a three sections XML document
 - Prefix-lists
 - Filters
 - Policy

XML document (1/2)

Prefix- lists

```

- <root>
  <!--This is a resolved XML policy file for AS199664-->
  <datetime>2016-05-20 14:12:00.938111</datetime>
  - <prefix-lists>
    - <prefix-list name="AS199664">
      <prefix type="ROUTE">185.49.140.0/22</prefix>
      <prefix type="ROUTE6">2a04:b900::/29</prefix>
    </prefix-list>
  </prefix-lists>

```

Filters

```

  - <peering-filters>
    - <peering-filter afi="IPV4.UNICAST" hash-value="d512041e8cf79a1c">
      <expression>AS199664</expression>
      - <statements>
        - <statement order="0" type="accept">
          <prefix-list>AS199664</prefix-list>
        </statement>
      </statements>
    </peering-filter>
    - <peering-filter afi="IPV4.UNICAST" hash-value="e97682b776fd9239">
      <expression>ANY</expression>
      - <statements>
        <statement order="0" type="accept"/>
      </statements>
    </peering-filter>
  </peering-filters>

```

XML document (2/2)

Peering AS

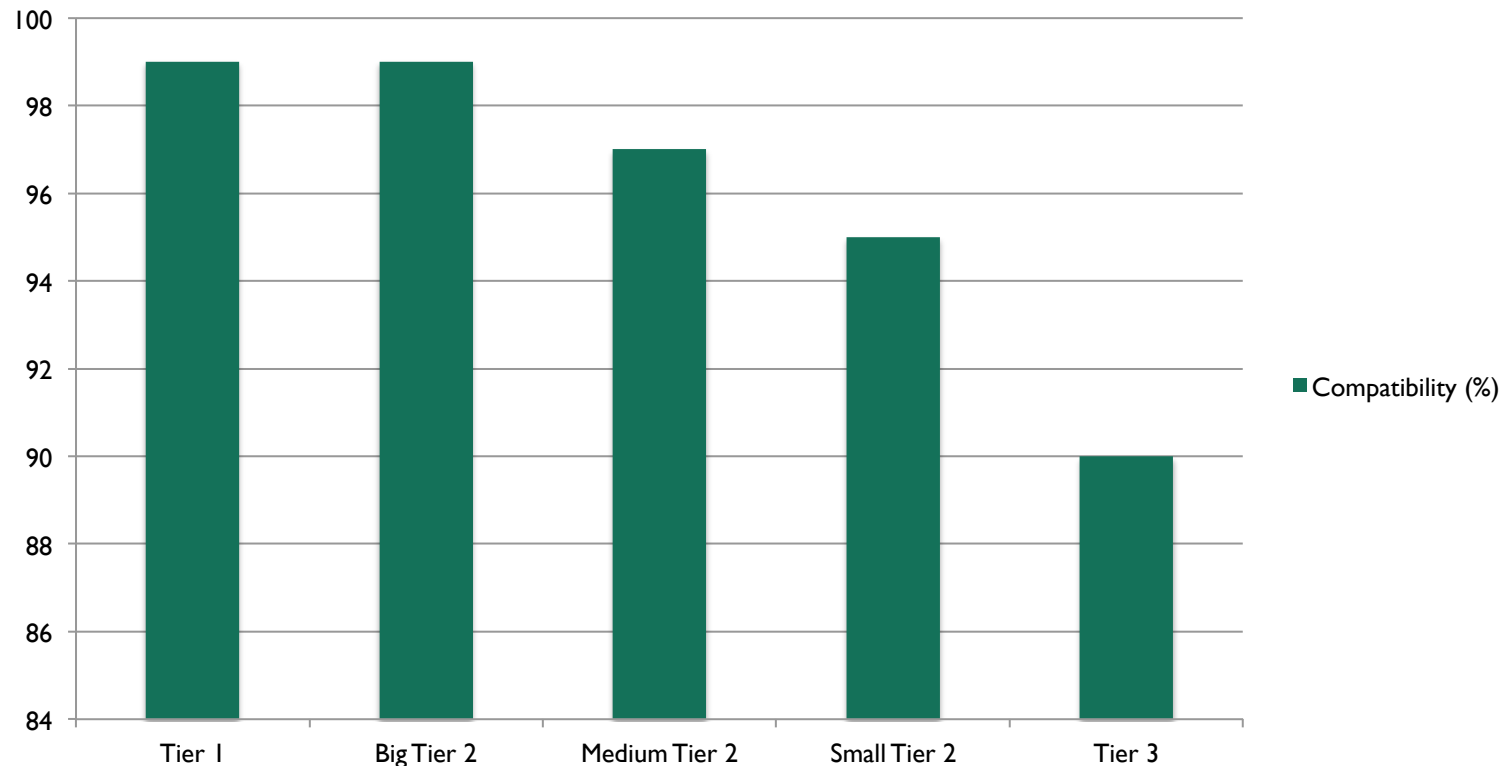
```

- <peering-policy>
+ <peer aut-num="AS43821"></peer>
- <peer aut-num="AS1257">
- <peering-points>
- <peering-point>
- <point local-IP="80.249.208.68" remote-IP="80.249.209.139">
  <actions_in pref="100"/>
  <actions_out/>
  </point>
</peering-point>
- <peering-point>
- <point local-IP="80.249.208.71" remote-IP="80.249.209.139">
  <actions_in pref="100"/>
  <actions_out/>
  </point>
</peering-point>
</peering-points>
- <imports>
- <filters>
  <filter>e97682b776fd9239</filter>
</filters>
</imports>
- <exports>
- <filters>
  <filter>1379c2192d65ecc5</filter>
</filters>
</exports>
<mp-imports/>
<mp-exports/>
</peer>

```

Filters

Parser compatibility



TIER classification based on CAIDA's ranking

Unsupported

- Multiple imports/exports per line
- PeerAS/FLTR-SER/RTR-SET
- Refine/Except
- Logical OR in AS-PATH <AS1 | AS2>
- Range operators on anything other than prefixes
- Nested operations with NOT
- 2 or more depth of operations

Parser performance

==== AS6799 ====

ASes: 234 (46)
AS sets: 42 (21)
RS sets: 3 (1)

Time ~ 0:03

==== AS6679 ====

ASes: 16636 (7)
AS sets: 4356 (20)
RS sets: 0

Time ~ 2:00

==== AS1103 ====

ASes: 33896 (81)
AS sets: 7079 (188)
RS sets: 3 (1)

Time ~ 4:30

==== AS286 ====

ASes: 45828 (189)
AS sets: 10259 (286)
RS sets: 0

Time ~ 6:00

Near future goals

- Implement a complete prototype
 - End-to-end configuration
- BGP filters only
- Juniper compatible

Deep future goals

- BGP filters and peers
- More vendor independent
- Integration with RPKI validator
- RDL*

* Remember RIPE 68: RDL by Benno

Participate?

- Experience
- Real life use cases
- Ideas
- Debugging

Thank you

- Acknowledgements
 - George Thessalonikefs (NLnet Labs)
 - Tomas Hlavacek (CZ.nic)