RPSL and rpsltool

Automatic generation of BGP configurations and filters

Marco d'Itri
<md@linux.it>
@rfc1036

Trex Workshop 2012 - September 14, 2012

Content

We will learn how RPSL works and how to use `rpsltool` to generate complete BGP configurations.

`rpsltool`: what for

Generates the configuration for BGP sessions

- customers
- peers
- IX route servers

It can generate filters from IRR data (i.e. the RIPE whois database).

What is RPSL

Routing Policy Specification Language

Is a language which allows an autonomous system to describe their routing policy in detail and use it to generate the matching configurations of routers.

Defined by RFC 2622 (1999) and others.

RPSLng (RFC 4012, 2005) extends it to support multicast and IPv6 neighbors.

RPSL is complex

Defined objects:

- `mntner, person, role`
- `aut-num, route, inet-rtr, filter, peering`
- `as-set, route-set, rtr-set, filter-set, peering-set`

Please raise your hand if you have ever seen a `rtr-set` object.

Almost all of these objects can be ignored in practice.

RPSL is complex (2)

Complexity extends to apparently familiar objects too.

Lesser known attributes of the `route` object:

- `components`
- `aggr-mtd`
- `aggr-bndry`
- `inject`
- `holes`
- `export-comps`

Please raise your hand if you have ever seen one in the wild.

RPSL is complex (3)

Fully describing a real world configuration in an `aut-num` object requires many directives (if possible at all!).

```
import:    { # MIX-IT peers
             from AS-ANY at 217.29.66.86
               accept not ({0.0.0.0/0^25-32}
                        or fltr-bogons-seeweb-it);
               action pref = 750; med = 0;
                      community = {12637:65002};
           } refine {
             from AS137 accept <AS-GARR+$>;
             # ...
           }
```

The `aut-num` object

They document the relationships among autonomous systems and the routes exchanged by them.

```
aut-num:        AS12637
import:         ...
export:         ...
```

Their purpose is to provide information which allow to configure your own router, but almost nobody uses them this way.

For third parties they only have information value: you should either keep them up to date or keep them as simple as possible.

The `route` object

A single route and the autonomous system which announces it:

```
route:          37.9.239.0/24
origin:         AS12637
```

The `route6` object describes IPv6 routes.

A side topic: the `pingable` attribute

```
route:          37.9.239.0/24
descr:          Seeweb s.r.l.
origin:         AS12637
pingable:       37.9.239.1
```

Defined by RFC 5943 (2010), it allows to publish an IP address which can be used by third parties for testing.

The `as-set` object

A list of autonomous systems:

```
as-set:         AS12637:AS-CUSTOMERS
descr:          Seeweb and its IPv4 customers
members:        AS12637, AS31076, AS6831, AS50627
members:        AS12654 # RIPE RIS Routing Beacons
```

Why I wrote `rpsltool`

IRRToolSet used to be the standard (and only) tool.

In 2005 it was not maintained, did not compile on modern systems and when it did it often crashed or silently generated incomplete filters.

Nowadays it is in a much better shape and hosted by ISC.

IRRToolSet is also extremely complex and implements rarely used features.

Why I wrote `rpsltool` (2)

I am lazy

- Configuring and maintaining peerings takes time.
- Boring tasks cause mistakes.

IRRToolSet vs. rpsltool: technology

IRRToolSet

- 26000 lines of C++ (used to be 70000)
- 4000 lines of flex and YACC parsers

rpsltool

- 1300 lines of perl
- 600 lines of Template::Toolkit

IRRToolSet vs. rpsltool: features

IRRToolSet allows to generate the complete BGP configuration for a network by defining it in the IRR.

This is complex enough that only an handful of networks in the world do it.

rpsltool is leaner and still it is easier to use for the common case.

Also, do you really want to publically advertise your complete routing policy?

IRRToolSet vs. rpsltool: IRR support

IRRToolSet

Uses all RPSL objects and attributes.

rpsltool

Only uses `route`, `route6`, `as-set`, `rs-set`.

rpsltool

Replaces the complex `aut-num` object with a simple list of neighbors.

rpsltool technology

perl

- YAML
- Template::Toolkit
- Net::Whois::RIPE

rpsltool targets

Example templates are provided for:

- IOS
- JunOS
- BIRD
- OpenBGPD

You can easily implement any other format you need.

Who uses rpsltool?

Internet exchanges

- MIX-IT (OpenBGPD), MINAP (OpenBGPD and BIRD).
- Others are considering it.

Random ISPs

If you need to configure BGP sessions for peers and customers, then rpsltool is for you.

rpsltool: basic principles

Input

- A list of peers and the parameters which describe them.
- RPSL objects from the IRR (optional).
- Configuration files templates.

Output

- The BGP configuration for a router.

Configuration example: peers

```
- as: 137
  description: GARR
  import: AS-GARR
  ip: 194.116.96.25
  ipv4: { aslist: 37 }

- as: 33077
  description: F root
  import: 2001:500::/48
  ip: 2001:7F8:23:FFFF::6
  backupip: 2001:7F8:23:FFFF::7
  bgp_commands: 'password 7 00000000000000000000'
```

This replaces the aut-num objects.

Configuration example: a template fragment

```
...
[% NEXT IF NOT neigh.$afi.import_routes.size %]
no [% acltype %] prefix-list [% aclname +%]
[% FOREACH route = neigh.$afi.import_routes %]
[% acltype %] prefix-list [% aclname %] permit \
                          [% route.route2cisco +%]
[% END %]
...
```

Templating allows to define every detail of the generated configuration.

Configuration example: not just config files

```
[%
   asset = 'AS12637:AS-CUSTOMERS';
   FOREACH as = asset.expand_as_set.asnsort;
     "# " _ as _ "\n";
     as.v4routes.aggregate.ipsort.join("\n");
   END
%]
```

Which objects should I maintain in the IRR?

The short answer is:

- `route`
- `route6`

Why bother registering my routes?

- Third parties will be able to automatically generate the filters for the BGP sessions with you.
- Some of your transit providers will be able to automatically generate filters for the BGP sessions with you.

Do I need to maintain my `aut-num` object?

The short answer is:

No.

But if you do not, at least try to not keep stale data there:

```
aut-num:      AS12637
import:       from AS-ANY accept ANY
export:       to   AS-ANY announce AS12637:AS-CUSTOMERS
remarks:      This is just an approximation.
```

(Not my idea, I stole it from DTAG.)

RPSL security

Creation of `route` objects is authenticated

A very complex set of rules guarantees that only the entity to which a network or ASN have been allocated to can create route objects referencing them.

Your customers' `aut-num` object may be useful as well

Some transit providers consider a routing policy described in an third party's `aut-num` object proof that you can announce their routes. This is very convenient and replaces LOAs.

But this is only valid in the RIPE region, registration in other IRRs is usually free for all!

Why filter peering sessions?

Is `maximum-prefix` enough?

It is a risk, and it causes downtime.

Is filtering always practical?

No, but you can do your best.

Questions?

`http://www.linux.it/~md/text/rpsltool-trex.pdf`

(google . . . Marco d'Itri . . . I feel lucky)