IRR Power Tools – A utility for managing Internet Routing Registry (IRR) filters.

By:

Richard A Steenbergen <ras@nlayer.net> nLayer Communications, Inc.

IRR – A quick review

- A loose collection of databases which allow BGP speaking networks to document their routes and policies for the purpose of automated router configurations.
- Currently speaks a language known as RPSL, with some extensions for v6/multicast known as RPSLng.
- A wide variety of databases exist:
  - Generic supported commercial-services: RADB
  - Generic unsupported free-services: ALTDB
  - Provider specific services: VERIO LEVEL3 SAVVIS etc
  - RIR-run databases: ARIN RIPE APNIC etc

Why IRR is better than manual filtering

- Almost too many to enumerate:
  - 1. Manually maintaining filters burns staff time and resources, both on the provider and customer network.
  - 2. The manual update process is often quite slow. This is especially true if the end customer buys transit from someone who buys transit from someone who buys transit…
  - 3. Humans make mistakes, and a typo in a prefix filter can result in customer outages (over-filtering) or leaks (under-filtering).
  - 4. Filtering a 10 route customer manually may be trivial, but filtering a 1,000 or 10,000 route customer whose prefix list changes by a handful of routes twice a day isn't. Many networks get lazy and switch large customers to "prefix-limit only" filters.
  - 5. Filtering peers manually has already proven to be more work than anyone is willing to do.
  - 6. It only gets harder from here, and the global routing table continues to grow.

So why doesn't everyone use IRR?

- Actually, a lot of people do:
  - IRR use among Europeans is almost total.
    - RIPE integrates IRR records into the IP allocation process.
    - Most providers insist on IRR records in order to provide BGP.
  - A few major networks in the US use IRR too:
    - NTT/Verio (AS2914)
    - Level(3) (AS3356)
    - SAVVIS (AS3561)
- But, most people don't:
  - UUNet, Global Crossing, AboveNet, Sprintlink, Cogent, AT&T, Qwest, Teleglobe, etc, etc, etc.

So WHY doesn't everybody use IRR?

- In an informal survey of US operators, they said:
  - The system is considered too complex, lacks real examples.
    - "For every 1 BGP speaking customer who uses IRR, there are 50 BGP speaking customers who have no hope of figuring it out, and they're going to go elsewhere if we try to make them."
    - "Look at our AS-SET, we can't even figure it out. Do you think our customers have a shot in hell of getting it right?."
  - Support costs actually go up if you have to teach your customers how to use IRR. Manual filters have no learning curve.
  - Most networks aren't interested in providing a public list of their customers and peers for IRR import/export policies.
  - Even among networks with strong IRR support, most still have to proxy register routes for their customers who just don't get it.
  - The existing tools are considered insufficient to actually deploy IRR filters in production, not stable and not enough features.

Well, at least we can fix one of those.

- What this tool does:
  - Automatic retrieval of prefixes behind an IRR object.
  - Automatic filtering of bogon or other undesirable routes.
  - Automatic aggregation of prefixes to reduce config size.
  - Tracking and long-term recording of prefix changes.
  - E-mails the customer and ISP with prefix changes.
  - Exports the change data to plain-text format, for easy interaction with non-IRR enabled networks.
  - Generates router configurations for easy deployment.

The pragmatic approach: What we don't do

- Import or export policies of any kind
  - Too complex, not needed for what we do.
- Filter-set, rtr-set, peering-set, inet-rtr
- AS-PATH filters
  - These depend on import/export policies to be effective.
- Any of that other funky stuff!

How is this different from IRRToolSet?

- This tool was originally built around IRRToolSet.
  - This was later rebuilt to bypass IRRToolSet, after getting feedback from dozens of network engineers who couldn't manage to compile IRRToolSet.
  - This is not a complete replacement, IRRToolSet is still far more powerful for generic IRR functions.
- IRR Power Tools builds upon simple queries, and adds features necessary for production use as a prefix-list manager.

Example Usage:

IRRDB.CONF:

```
#########################################################
#ASN    IRR Object                          Update Email
#########################################################
1234    AS-EXAMPLEOBJECT1 noc@examplecustomerabc.com
2345    AS-EXAMPLEOBJECT2 noc@anotherexamplecust.com
3456    AS3456            noc@angrignonheartnina.com
```

EXCLUSIONS.CONF: Bogons and other filters

IRRPT.CONF: General configuration, aggregation, etc.

NAG.CONF: E-Mailing providers who don't speak IRR.

Example Usage: Fetching New Prefixes

```
$ ./irrpt_fetch
Processing AS812 (Record 1)
   - Importing /usr/local/irrpt/db/812          version 1.1
   - Importing /usr/local/irrpt/db/812.agg      version 1.1
   - Sending update notification to noc@angrignon.com
Processing AS8001 (Record 2)
   - Updating /usr/local/irrpt/db/8001    version 1.1 -> 1.2
   - Updating /usr/local/irrpt/db/8001.agg version 1.1 -> 1.2
   - Sending update notification to eng@nina.net
Completed processing of 2 IRR object(s).
```

Example Usage: Prefix database

```
-rw-r--r--  1 user  group  49197 Nov 24 03:39 8001
-rw-r--r--  1 user  group  19086 Nov 24 03:39 8001.agg
-rw-r--r--  1 user  group   5520 Nov 24 03:39 812
-rw-r--r--  1 user  group    366 Nov 24 03:39 812.agg

$ wc -l db/8001
    3149 8001
$ wc -l db/8001.agg
    1213 8001.agg
$ wc -l db/812
     379 812
$ wc -l db/812.agg
      26 812.agg
```

Example Usage: E-Mail Updates

```
From: eng@yourcompany.com
Subject: [IRRPT] Changes to AS8001 (Aggregated)

Remove 8.9.3.0/24
Remove 8.9.4.0/23
Remove 24.228.0.0/18
Add 4.17.225.0/24
Add 12.26.83.0/24
Add 12.31.6.0/24

Complete list for AS8001 (object AS-NAC) (Aggregated):
4.17.225.0/24
4.17.226.0/23
4.17.251.0/24
...
```

Example Usage: Prefix-list Generation

```
$ ./irrpt_pfxgen 8001
conf t
no ip prefix-list CUSTOMER:8001
ip prefix-list CUSTOMER:8001 permit 4.17.225.0/24
ip prefix-list CUSTOMER:8001 permit 4.17.226.0/23 le 24
ip prefix-list CUSTOMER:8001 permit 4.17.251.0/24
ip prefix-list CUSTOMER:8001 permit 4.17.252.0/23 le 24

$ ./irrpt_pfxgen -f juniper 8001
policy-options {
    replace: policy-statement CUSTOMER:8001 {
        term prefixes {
            from {
                route-filter 4.17.225.0/24 upto /24;
                route-filter 4.17.226.0/23 upto /24;
…
```

Example Usage: Nagging Providers

```
Hello,

This is an automated e-mail message requesting a prefix list
   modification from providers who still build their prefix filters
   manually, rather than from Internet Routing Registry (IRR)
   databases.

A list of requested changes is contained below, followed by a
   complete list of prefixes for your reference. Also please note
   that the prefixes are pre-aggregated, and should always be applied
   with at least "upto /24" (Juniper) or “le 24” (Cisco).

Requesting Company.......................... Your Company Name, Inc.
Requesting Autonomous System Number (ASN)... 12345
Customer Identifying Information............ Customer Identifier 1
Routine / Successful Completion Notices..... ops-list@company.com
Failures, Problems, Questions, Etc.......... ops-alert@company.com
```

Where can I get it?

- Sourceforge:
  - http://irrpt.sourceforge.net

Future Development Plans

- Add support for IPv6/RPSLng
  - Needs IPv6 aggregation tools
- Add support for RIPE-protocol whois queries
  - Not as efficient as RADB-style query protocol, but needed to query route6 objects.
- Add SQL database support for a backend.
- Convert from a script to a real application.
- IRRWeb – http://www.irrweb.com/

Send questions, complaints, to:

Richard A Steenbergen <ras@nlayer.net>

Happy Valentines Day!