Peering 101 Handbook

Release 0.1

Matthew Walster

December 30, 2015

This document is under construction.

Contents:

CHAPTER 1

Peering vs. Transit

When you took your transit connections, you probably just accepted every prefix they sent you. You might have applied some policies to it such as reducing the local preference or accepting/ignoring MEDs, but for the most part you're just receiving BGP UPDATEs from them and installing them into your routers FIB. You trust your transit provider to only send you prefixes that not only they can reach, but also that they have validated as belonging to their customers – or routes through to other transit providers that have done similar due diligence.

However, you're now looking to peer with another network. If you set it up identically to a transit provider, the traffic will flow out of that peering connection assuming the AS Path is shorter or some other locally set metric. Unfortunately, that peer may advertise (accidentally or nefariously) prefixes to you that they do not own, either as another AS along the path or even worse as the origin for those prefixes! Your router will blindly accept these prefixes and route your traffic to their AS which may end up with the traffic being dropped due to congestion or attack.

Let's compare this with any downstream customers you may have: You probably had a prefix list (manually edited) to only accept certain prefixes from them. If they do leak a full transit feed to you, you're just going to ignore them. However, although you're ignoring the prefix announcements they may all be stored in memory and therefore using precious router resources.

What's needed is a way to verify that not only are your peers sending you prefixes they have been delegated by the RIRs, but that any downstream customers they have are being advertised on with a valid AS Path.

This document seeks to show you how to generate (and hopefully automate) prefix lists, AS Path filters, and a few other more recent tools that will allow you to verify that these resources have been advertised to you with the consent of the RIRs involved. This will allow you not only to become a "good netizen" but also prevent you from being affected by many of the routing errors the community has seen in recent times.

In short: These processes will greatly help you avoid incidents of your peers causing outages on your network.

CHAPTER 2

Setting a “max-prefix” limit

Setting a limit on the number of prefixes you’ll accept from a peer is one of the simplest things you can do. What this does is say “if you send me more than X prefixes, I will forcibly bring down my peering with you”.

You will see recommendations of various algorithms to set these prefix limits – normally 10% of the number of prefixes you expect from them. The problem is that when you set these limits it is easy to forget they are in place and when they take a new customer on you may see a sudden jump in the number of prefixes, causing your session to reset.

The purpose of this limit is a “final” failsafe. If your prefix lists fail, you want this to bring down the session which will hopefully alert both your NOC and the peer that something “incorrect” has happened. Therefore, it wants to be high to prevent accidentally tripping it but also low enough to not blindly accept them accidentally sending you all their peering routes etc.

The suggestion here is to multiply the number of routes they will be sending you and multiply it by 10:

Received	Limit
1	10
35	350
950	9500
75000	500000

One peer is sending us 75000 prefixes, yet the limit table says 500000 – at the time of writing the DFZ is reaching 600000 prefixes. If they send more than 500000 prefixes we are almost certainly receiving a full table from them so we need to drop it before we reach that number. Very few peers will ever send you this many prefixes, and indeed your routing platform may not accept this many prefixes in a prefix list so you may have to rely on just this rudimentary way of detecting a routing mistake.

The max-prefix limit is usually set per-neighbor. A Cisco IOS router will be configured in the following way:

```
neighbor 10.1.1.1 maximum-prefix 350
```

This will drop the peer 10.1.1.1 when more than 350 prefixes are received.

There are some other options you can consider:

```
neighbor 10.1.1.1 maximum-prefix 350 warning-only
```

This will only warn when the peer sends more than 350 prefixes, the session will remain Established.

```
neighbor 10.1.1.1 maximum-prefix 350 100
```

This will warn at 100 prefixes (remember, you’re expecting 35) and drop at 350.

```
neighbor 10.1.1.1 maximum-prefix 350 100 warning-only
```

This will warn at 100 prefixes as before, but send a different warning at 350. The session will remain Established.

Similar functionality is present in other network operating systems. Full examples for your device may be posted at the end of this document – if it is missing and you're unsure how to proceed, please do feel free to ask and a later revision of the document may include your router syntax!

Generating a Prefix List for your peer

A prefix list applied to a BGP peer allows your BGP speaker to only accept a certain set of prefixes on that BGP session. For instance, if you have a prefix list containing “192.0.2.0/24” and the peer tries to advertise “198.51.100.0/24” to you, it will reject that prefix. It will not tear down the BGP session however.

A potential “gotcha” is that if you are only accepting “192.0.2.0/24”, you can’t accept “192.0.2.0/25” because that is not an exact match. Your router operating system may have syntax so allow this, such as “192.0.2.0/24 ge 25 le 25” which will allow any prefix in “192.0.2.0/24” with CIDR mask “/25”, i.e. “192.0.2.0/25” and “192.0.2.128/25”. Note that this does NOT include “192.0.2.0/24” itself so be careful with your syntax.

There is an easy solution to this problem, however. A tool called *bgpq3<https://github.com/snar/bgpq3>* exists which automates all this for you.

Firstly, a couple of concepts you need to be aware of:

If you peer with AS65000, they may just be sending you prefixes from AS65000 only.

If you peer with AS64512, they may also be sending you prefixes from AS64512, and all their customers, AS65001 AS65002 AS65003 etc.

To make life easier, *RPSL<http://www.irr.net/docs/rfc4012.txt>* specifies an “as-set” – a set of ASNs in one object. It looks like this:

```
as-set:         AS-EXAMPLE
descr:          Example AS-SET
members:        AS64512
members:        AS65001
members:        AS65002
members:        AS65003
tech-c:         EXAMPLE-RIPE
admin-c:        EXAMPLE-RIPE
mnt-by:         EXAMPLE-MNT
changed:        noc@example.net 20150712
created:        2015-07-12T14:45:58Z
last-modified:  2015-07-12T20:20:01Z
source:         RIPE
```

We can see here that if we expand “AS-EXAMPLE”, we will get AS64512 plus all it’s downstream customers. A “members:” entry may contain a further as-set, and can easily be recursed by bgpq3.

We can then find out which “route:” objects are stored in the IRRDB by performing an inverse lookup, such as:

```
whois -i origin AS65000
```

This gives you many route objects that this ASN has registered it is “origin” for. This is something you can parse and create a prefix list with – do not do this. You will get it wrong and find a corner case that will break your system. Use bgpq3 – it's really good at this!

Let's have a look at a sample bgpq3 run using Cisco IOS (bgpq3 currently supports IOS, IOS XR, JUNOS, BIRD – more can be supported if demanded):

```
$ bgpq3 AS65000
no ip prefix-list NN
ip prefix-list NN permit 192.0.2.0/24
ip prefix-list NN permit 192.0.2.0/26
ip prefix-list NN permit 192.0.2.64/26
ip prefix-list NN permit 192.0.2.128/26
ip prefix-list NN permit 192.0.2.192/26
```

We can see here that AS65000 has registered it will be advertising three prefixes, and bgpq3 has spit out some configuration that will get rid of prefix-list "NN" and replace it with the three new prefixes. That's a bit wasteful though – five entries. I'm sure we can summarise this... bgpq3 will do it for you!

```
$ bgpq3 -A AS65000
no ip prefix-list NN
ip prefix-list NN permit 192.0.2.0/24
ip prefix-list NN permit 192.0.2.0/24
ip prefix-list NN permit 192.0.2.0/24 ge 26 le 26
```

Excellent. However, if I now run it for AS65001 it will be over-written. Let's put a name in:

```
$ bgpq3 -A -l EXAMPLE-in AS65000
no ip prefix-list EXAMPLE-in
ip prefix-list EXAMPLE-in permit 192.0.2.0/24
ip prefix-list EXAMPLE-in permit 192.0.2.0/24 ge 26 le 26
```

We don't want to accept any prefixes longer than a /24, as those are generally filtered on the internet:

```
$ bgpq3 -A -l EXAMPLE-in -m 24 AS65000
no ip prefix-list EXAMPLE-in
ip prefix-list EXAMPLE-in permit 192.0.2.0/24
```

Perfect. To summarise:

Option	Function
-A	Aggregates prefixes as much as possible
-l	Labels the prefix list with a friendly name
-m	Maximum Size of prefix to be accepted

With this output, you can install these prefix lists onto your router with your favourite tool (or do it by hand, but preferably do this at least once a week at a minimum – most providers do this once per day)

CHAPTER 4

Validating AS-PATH attributes

Verifying the prefix origin is registered through this peer is a good start, but there are other attributes we can validate also – the major one being the AS Path.

With bgpq3, this is as simple as using the "-f" option which generates an as-path filter for you for a given AS-SET:

```
$ bgpq3 -f 65000 AS65000
no ip as-path access-list NN
ip as-path access-list NN permit ^65000(_65000)*$
```

You will now only accept prefixes with the AS Path attribute starting "65000" and ending "65000", with any number of "65000" in between (this is in case the network is prepending to you, there may be multiples of their ASN).

However, if we try and do this for an as-set, we'll see more data:

```
$ bgpq3 -f 64512 AS-EXAMPLE
no ip as-path access-list NN
ip as-path access-list NN permit ^64512(_64512)*$
ip as-path access-list NN permit ^64512(_[0-9]+)*_(65001|65002|65003)$
```

Here we can see that we'll accept anything with the next ASN in the path being any number of "64512" ASNs, followed by a path terminating in AS65001, AS65002, or AS65003. This was expanded form the AS-SET "AS-EXAMPLE" which has these ASNs as members.

This access-list can then be referred to as usual within your route-map:

```
router bgp 65000
  neighbor 10.1.1.1 remote-as 64512
  neighbor 10.1.1.1 route-map EXAMPLE-in in

route-map EXAMPLE-in permit 10
  match as-path NN
```

Using Regular Expressions such as those in the as-path access-lists above may cause strain on your router. While they may be of benefit, many networks opt not to implement them because it greatly increases convergence times on older platforms. Remember that these filters are only evaluated on the prefix being selected for consideration – you should have a backup route to this prefix, usually transit. Therefore, while your router may take a little time to flip traffic onto the peering path it is probably worth the wait to validate the IRRDB approves of the path.

Using as-path access-lists is NOT RECOMMENDED if you do not have a 32-bit ASN capable router, or the peer is likely to have "AS23456" (the 4 byte transition ASN) in the path. This was more of a problem 5-10 years ago, but certain platforms still do not have good 32-bit ASN support. It is an exercise left to the reader to determine if their router operating system has sufficiently new code to perform this operation. Anything from 2015 or later is likely to be a direct "OK to use" answer from your support partner.

CHAPTER 5

RPKI

For routers that support it, how do we utilize the RPKI to validate that the person originating the prefix is authorized to do so?

This obviously does not help with accidentally routing stuff through an ASN, but will help combat prefix hijacking etc.

CHAPTER 6

Full Example

An example of a fully worked through configuration.

6.1 Cisco IOS

When setting up the peer:

```
router bgp 65000
  neighbor 10.1.1.1 remote-as 64512
  neighbor 10.1.1.1 prefix-list EXAMPLE-in in
  neighbor 10.1.1.1 prefix-list AS65000 out
  neighbor 10.1.1.1 route-map EXAMPLE-in in
  neighbor 10.1.1.1 route-map AS65000 out
route-map EXAMPLE-in permit 10
  match as-path EXAMPLE-in
```

Automatically generated by bgpq3, once per day:

```
no ip prefix-list EXAMPLE-in
  ip prefix-list EXAMPLE-in permit 192.0.2.0/24
no ip as-path access-list EXAMPLE-in
  ip as-path access-list EXAMPLE-in permit ^64512(_64512)*$
  ip as-path access-list EXAMPLE-in permit ^64512(_[0-9]+)*_(65001|65002|65003)$
clear ip bgp 10.1.1.1 soft
```

Note that “soft” will do a route-refresh is supported, or soft-reconfiguration if configured. It will not reset the session.

6.2 Cisco IOS-XR

Once IOS finished, re-run bgpq3 with “-X” – note that only prefix lists are supported.

6.3 Juniper JUNOS

Once IOS finished, re-run bgpq3 with “-J” for both prefix lists and as-sets.

6.4 Brocade IronWare

While very Cisco like, features will need to be tested before posting any code here.

6.5 Arista EOS

While very Cisco like, features will need to be tested before posting any code here.

6.6 BIRD

Once IOS finished, re-run bgpq3 with "-b" for both prefix lists and as-sets.

and more...

Further Reading

Sources used in the construction of this document, or material you may find helpful in reading more into this topic include:

RFC7454 (aka BGP OPSEC)

How to contribute:

- Fork https://github.com/dotwaffle/peering101
- Make changes – you'll want to use http://sphinx-doc.org/rest.html
- Push them to GitHub
- Make a Pull Request
- Once merged, the document updates automatically!

How to point our errors or missing information:

- Submit an issue at https://github.com/dotwaffle/peering101/issues
- We'll email you if there's any further clarification needed!