June 2024
Geoff Huston

Routing Topics at RIPE 88

RIPE 88 was held in May 2024 at Krakow, Poland. Here's as summary of some of the routing topics that were presented at that meeting that I found to be of interest.

Bgpipe – A BGP Reverse Proxy

Observing and measuring the dynamic behaviour of BGP has used a small set of tools for quite some time. There's the BGP Monitoring Protocol (BMP, RFC 7854), there's the Multi-threaded Routing Toolkit (MRT) for BGP snapshot and update logs, and if you really want to head back to earliest days of this work, there are scripts to interrogate a router via the CLI. All of these are observation tools, but they cannot alter the BGP messages that are being passed between BGP speakers.

The bgpipe tool, presented by Pawel Foremski, is an interesting tool that operates both as a BGP "wire sniffer" but also allows BGP messages to be altered on the fly (Figure 1).

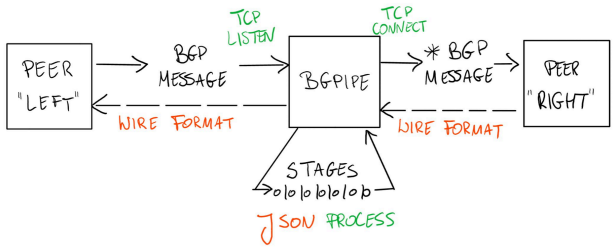

Figure 1 – bgpipe overview (from RIPE 88 presentation)

Internally, the bgpipe process can be configured to invoke supplied "callback" routines when part of a BGP message matches some provided pattern, such as a particular IP prefix, update attribute patters or such, and it can also be configured to have "events" which processing elements in bgpipe can subscribe to. Simple use cases are to take a BGP session and produce a JSON-formatted log of all BGP messages, or taking an unencrypted BGP session and adding TCP-MD5 encryption. More advanced cases can make use of an external call interface to add Route Validation checks using RPKI credentials.

There has been some concern about using IPv6 prefixes to perform a BGP more specific route flooding attack and its possible to use a bgpipe module to perform various forms of prefix thresholds (per origin AS or per aggregate prefix) to detect and filter out the effects of such flooding attacks.

It's early days in this work, but it is certainly an intriguing and novel BGP tool (https://bgpipe.org).

Presentation link.

Reclaiming Class E Space

In the Internet's early days the IPv4 address space was divided into "Classes," with Classes A through E, as determined by the value of the first octet of the address.

To briefly recap for those who came in late, Class A space was unicast space where the first octet of the IP address was a network identifier, and the remaining 24 bits identified a host's attachment to this network. Class A networks spanned the address range from 1.0.0.0 through to 126.255.255.255. or just under one half of the total IPv4 address span. Class B space used a 16-bit network identifier value and spanned the address range 128.0.0.0 through to 191.255.255.255, or one quarter of the IPv4 address span. Class C space used a 24-bit network identifier and spanned the address range 192.0.0.0 through to 223.255.255.255, or one eighth of the IPv4 address span. Class D was intended for multicast use and used the address range of 224.0.0.0 through to 239.255.255.255, or one sixteenth of the address span.

That plan left the address span from 240.0.0.0 through to 255.255.255.255, which was termed "Class E" address space and was annotated in the IANA registry as "Reserved for Future Use".

By 2005 or so, when the exhaustion of the IPv4 address pool was looking more and more imminent it appeared to be odd to be running out of IPv4 addresses while one sixteenth of the address space was still designated as being reserved for "Future Use". Surely the usable lifetime of IPv4 was drawing to a close, so the "future" envisioned by this reservation was in fact "now". An internet draft from this time (draft-wilson-class-e) advocated the redesignation of this address black as unicast private use.

The problem was that many implementations of the IP protocol stack, used in both hosts and network equipment discarded packets that had had Class E addresses in the IP packet header. There was a view at the time that the effort to clear the various Class E blocks in network equipment and hosts that prevented the use of this Class E address space would be better spent on the IPv6 transition, and the topic of redesignating this IPv4 space for use petered out.

The conversation appears to have returned in recent years and a presentation by Ben Cartright-Cox and Zybnek Pospichal outlined some recent tests on routing equipment when attempting to configure a BGP peering using addresses from this Class E space. Their results indicate that blocks on the use of this address space still remain in some vendor's equipment.

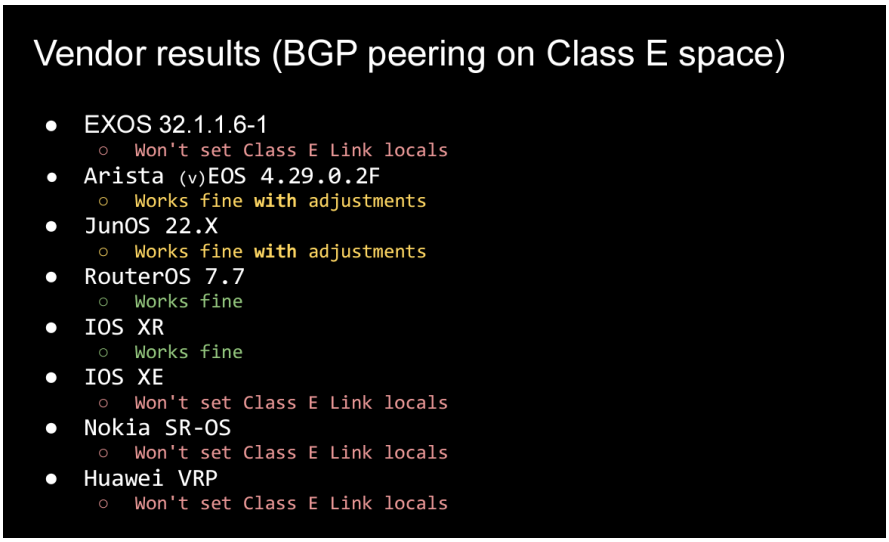

Figure 2 – Router Treatment of Class E Space (from RIPE 88 presentation)

They also describe the results in attempting to configure a network using Class E space, again with varying results. Is the pain and effort of chasing down the issues with supporting the use of Class E in a network worth the potential gain of increasing the pool of additional IPv4 private use space? Or would just deploying IPv6, using either global unicast address space or ULAs be a lower pain path with a more assured longer-term future?

Presentation link

RPKI Validation Reconsidered Reconsidered

In 2018 the IETF published RFC 8360, proposing a change in the validation algorithm for RPKI certificates. The observation was that inconsistencies between a certificate and a subordinate certificate product would invalidate the entire tree of subordinate certificates that are descendants of the errant certificate. For RPKI operators that operate high in the RPKI certificate hierarchy this risk poses a rigid requirement for operational excellence at all times. There is simply no margin for error here, and changes in resources holdings in these certificates must be promulgated in strict order. The order is necessarily "bottom up" in the case of removing resources from a resource set, and top down in the case or adding resources to a set.

RFC 8360 proposed a change in the RPKI validation scheme that was more tolerant of certain forms of mismatch between the resource sets of an RPKI certificate and its subordinate products. However, the edict at the time from the X.509 folk in the IETF was that a change in validation algorithm for such certificates required a change in the Object Identifier (OID) values of these RPKI certificates. The inability to allow a client to determine which validation approach it wished to use was considered to be outside the "proper scope" of X.509 certificate use (even though these same clients are free to choose their own trust anchors for validation). This implied requirement to operate two parallel RPKI certificate hierarchies while we all transitioned from one algorithm to the other was generally thought to be nonsensical, and the entire concept of the altered validation algorithm quietly died in a dusty corner!

Job Snijders and Ben Maddison would like to see the RFC 8360 algorithm revived. Their proposed path forward is to deprecate the OID allocation in RFC8630 and instead update RFC 6482 and RFC 6487 with the revised validation algorithm. In other words, change the standard profile of the definition of the validation algorithm and keep the OID value in RPKI certificates constant.

While it may look like taking the long way round to the same outcome, in many ways it's a sensible change. This form of validation improves the robustness of the RPKI system by a small set of changes to the certificate validation process. If we can avoid dancing around with obscure changes to OIDs to RPKI certificates and if we can avoid the cost of running parallel repository structures when transitioning from the current validation algorithm to this revised algorithm, then so much the better!

Presentation link.

Low Latency RPKI Validation

How "fast" should the RPKI system operate? How quickly should changes in the credential space of ROAs and RPKI certificates be reflected in changes to the route acceptance criteria used by RPKI-aware BGP speakers? A few hours perhaps? Seconds? Or maybe Milliseconds? The best answer I've heard to this question is "as fast as BGP route propagation". Any faster is just over-achieving!

The most obvious way of ensuring that the RPKI system is as responsive as BGP itself is to use BGP as the RPKI credential flooding platform. However, one major router vendor some two decades ago, when the RPKI effort was starting up, was sick and tired of incurring ongoing costs in adding adornments to their BGP implementation as a result of chasing down every wild idea about modifications to BGP behaviour that was aired in the IETF. One of the impositions imposed on the SIDR effort by one of the Routing Area Directors at the time (employed by said router vendor, coincidentally) was that it was not possible to propose changes to BGP when adding security mechanisms to the BGP protocol.

So, the SIDR effort followed the only other path that was available to it, namely, to use a separate credential publication system and a separate flooding protocol to pass these credentials to every RPKI-aware BGP speaker. The design of the RPKI credential system was simple. Each client maintained a local caches of all RPKI published objects by periodically polling every RPKI publication point, checking to see if anything had changed since the last poll. It's simple but some 40 years of operating distributed systems has taught us that information flooding using uncoordinated polling is horrendously inefficient under load, so this approach simply does not scale efficiently! As the number of publication points increases the load on each client increases, and as the number of clients increase then the load on each publication server increases. And the only way to make the system more responsive is to decrease the polling interval, which increases the load on both the publication servers and the relying party clients.

If we want a more efficient RPKI system, then increasing the frequency of polling is not a sensible scalable approach. This is pretty much the exact opposite of the message being conveyed in this presentation. Thrashing around with a polling system just to see if any published object has changed is not a sensible way of achieving coherency across a large set of local caches. And increasing the frequency of polling disproportionately increases systemic load with only marginal improvements in latency. Pushing changes through the system in response to notified changes is a far more scalable approach.

BGP has the property of being the most efficient flooding protocol we know of in the interdomain routing space, and the underlying approach used by BGP is a push-based approach passing through only the changes to the information base as and when they occur. Using BGP itself to perform this RPKI credential flooding, in much the same manner as the now almost 30-year old soBGP proposal, is as sensible today was it was back then!

Presentation link.

Revisiting BCP194

What does a secured routing environment look like? That's a deceptively hard question in many ways, in so far as it encompasses questions of what we are trying to secure and how. That was the objective of BCP 194 (or RFC 7454), BGP Operations and Security. The document describes measures to protect the BGP sessions from third party attempts to interfere with the session, using techniques such as Time to Live (TTL), the TCP Authentication Option (TCP-AO), and control-plane filtering. The document also describes measures to ensure the authenticity of routing information and controlling the flow of routing information, using prefix filtering, AS Path filtering, route flap dampening, and BGP community scrubbing.

The document was published in 2015 and its certainly not ageing well. It was a highly prescriptive document more like a shopping list than a general set of principles and practices, and the issue with such documents is that they quickly age out. An effort by Tobias Fiebig to create a revised version of this list made an already voluminous shopping list of security-related tasks even longer at 55 pages (draft-ietf-grow-bgpopsecupd). The problem with such an attempt at a comprehensive task list is that it is never complete, and the tasks tend to age quickly, so the maintenance load of such a document is significant. Perhaps a new approach is necessary.

Rather than attempt to spoon-feed a reader with a long list of detailed tasks, it is more useful to elevate the level of thinking and describe what these operational security measures are trying to achieve and leave it at that. There is a new revision being prepared (using github), and this revision of the draft has been stripped down to around 4 pages! Any effort to make such documents as simple and brief as they can be (but no simpler, of course) is to be applauded!

As Tobias points out, the basic messages in BGP security are:

- Make sure no funny packets get to your BGP speakers
- Make sure nobody meddles with your BGP sessions

- Don’t import anything you shouldn’t
- Don’t export anything you shouldn’t
- Don’t meddle with BGP data not meant for you

This seems like a very sensible set of guidelines for everyone!

Presentation link.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net