February 2025
Geoff Huston

BGP Zombies at NANOG 93

The internet is held together by the Border Gateway Protocol (BGP). It's an “interesting” protocol in many ways. Each BGP speaker uses a reliable transport protocol to send local reachability information to its adjacent peers. The inference is once a BGP neighbour has been informed about a new item, whether it’s an update about reachability of an address prefix, or its unreachability, then there is no need to repeat this information until the state of the prefix has changed. A BGP session running in a steady state only sends its neighbours information about what’s changing. The assumption is that for the lifetime of a BGP session the BGP engines have a “perfect” memory.

What happens when the BGP engine is not perfect in its memory? What happens if a BGP neighbour sends a withdrawal about a prefix and the local router manages to drop this information rather than process it? The basic answer is nothing of much consequence happens. The local BGP speaker still believes that the route exists, and if it receives a packet addressed to a destination within this address prefix, then it would forward the packet toward the router that it still believes lies on the path to this destination. Given that this adjacent router has already withdrawn this route, then the fate of any such packet is that it would be discarded, so the outcome is much the same. There are more esoteric situations where a more specific route is stuck in this manner and a covering aggregate route points in a different direction, and in this case traffic that is directed to this zombie route would be forwarded incorrectly.

In the early specifications of BGP (up to the publication of "Route Refresh", RFC 2918 in 2000) there was no way for a peer to request a “refresh” of its local copy of prefixes that have been sent by a peer (the “route refresh” function), so once a BGP speaker has managed to get out of sync with its neighbour, then the only way to remedy the situation was to tear down the BGP session and restart it. Even this is not exactly all that helpful, given that the sender of a withdrawn route assumes that the adjacent BGP speaker has received it and already acted upon it, and the receiver has managed to drop the transaction before it had got around to processing it, so it is unaware that it’s holding onto a route in error. So, the local BGP instance is not necessarily aware that it needs to issue a Route Refresh request in any case! The implication is that a conservative peer would issue such a route refresh message on a regular basis, but how frequently is an open question.

It was long suspected that BGP route entries might get stuck in routers in this manner, but it appears that this topic was not widely noted until Gert Doring started doing his regular report on the state of the IPv6 routing table at RIPE meetings in Europe in the early 2000's. For example, in his report to RIPE 42 in April 2002, Gert reported on an IPv6 "zombie” route he had found in the IPv6 routing table. At the time there were just some 300 IPv6 routes in total, so it was feasible to examine the entire route set manually and identify these stuck routes with some certainty!

It was unclear at the time if this was an isolated situation, or one that was related to the relatively little-used code paths of IPv6 support in routers at that time. Maybe it was just an IPv6 behaviour in BGP. However, there was also a larger question concerning the IPv4 routing table. In 2000 the IPv4 BGP routing table contained around 70,000 entries at the start of the year, and 102,000 by year's end. That's a lot of growth in just one year, and nobody was in a position to claim that they had looked at each and every route and could vouch that each and every route was current and valid. In which case nobody could answer the question: "How many of these IPv4 routes are "zombies"? Or, in other words, is this growth real, or a form of routing "chaff" caused by a defect in BGP implementations? The routing space was on track for another growth crisis, similar to the earlier routing panic in 1990-1991, and it would be useful to figure out if this growth was due to a side-effect of the Internet boom that was underway, or just the result of a poor BGP implementation.

The year 2000 was a special year, as it was at the peak of a bubble of irrational euphoria, and the bubble burst late in that year. In 2001 a less exuberant Internet appeared, and the growth in the routing table was essentially non-existent for the entire year. At this point the collective appetite to hunt down zombie routes in the routing table quickly dissipated, and the topic reverted to a curious anomaly that a just a few network engineers bothered to look at. The scale of these zombies was suspected to be extremely small; they appeared to be challenging to generate on demand for specific addresses in specific locations, and the implication was that they were not considered to be a threat to the integrity of the routing system. The topic of routing zombies was placed in the "curious, but mostly harmless" category of routing anomalies.

The topic of BGP zombies has resurfaced from time to time over the ensuing 25 years, and the most recent such resurfacing was this month at NANOG 93, in a presentation by Iliana Xygkou from ThousandEyes. It's still challenging to phrase the presence of BGP zombies as a security threat, but without really knowing how many of the one million or so routes in today's IPv4 BGP routing table are just zombie junk, then we might be too dismissive about zombies. But there is perhaps a more useful set of questions to be asked: Where are Zombie routes being generated? And who is generating them? And why are they generating Zombies?

Detecting BGP Zombies is challenging. If one had a universal view of the internal state of every BGP speaker, and one could cross-correlate these states then yes, it would be possible to define an algorithm that could detect these zombie anomalies. With a BGP version of Jeremy Bentham's Panopticon, then it would be possible to audit BGP routes to confirm that each route is present in the AS's that are sitting on the AS Path for every route. Where the routes exist only at the "end" of the AS Path, but not at and near the origination, then it would be possible to identify a zombie, and infer the point where the router got stuck, creating the zombie. Such a comprehensive view of the operation of BGP does not exist, and the best we've been able to come up with is a set of pulsing announcements that appear to trigger some BGP speakers into carrying a zombie routing state.

There are a number of so-called "Routing Beacons". These beacons announce and withdraw a route object on a 2-hour cycle. Not only do these beacons allow researchers to study the behaviour of BGP route propagation of announcements and withdrawals, but they open up the possibility of zombie hunting, looking for networks that are retaining the route during the 2-hour period when the route has been withdrawn.

Why do these zombies occur?

It may be possible with the various processing and scheduling paths within a router that a route withdrawal might get discarded before the cleanup functions within the router's data structures have completed. At this point the route would persist as a zombie until it was subsequently re-announced. In such scenarios the route set in the impacted BGP speaker would look quote normal, and its local FIB state would be synchronised against its peers, as is the intention of the BGP protocol, with the exception of these wedged persistent routes.

It is also possible that the TCP session between the two BGP speakers has jammed, and the receiver is blocking any further updates from the sending peer. This is described in detail by Ben Cartwright-Cox in his blog post on stuck BGP routes. The scenario described is where a BGP process ceases to accept further data on the TCP session with a peer. The local TCP input buffer will fill, and the TCP session will then advertise a 0-sized window to its peer, preventing the remote peer from sending further BGP messages. The local session is not allowing the remote BGP peer to send any further messages over BGP, but it can still send messages, including KeepAlive messages. That means that the remote session is still receiving regular BGP KeepAlive messages, so it will not unilaterally tear down the BGP session. This situation also assumes that the local BGP speaker with the jammed TCP state will be in a state where the cessation of the stream of regular KeepAlive messages does not cause the local BGP speaker to tear down the session, both at the BGP level and the TCP level.

If this is the case, then this calls for a more forceful response than a Route Reset request, as the TCP session itself needs to be torn down and restarted. Here the more recent SendHoldTimer (RFC 9687) setting might be of use. This is a recent (November 2024) addition to the BGP-4 protocol that requires a BGP instance to be able to query the underlying TCP session to determine if the connection is still sending data to the remote end.

When the local SendHold Timer expires without any BGP messages being sent to the remote peer, then the local BGP instance would shutdown the session and clear all locally held routes that are held in the local adjacency routing information structure (adj-RIB-in) associated with that remote peer.

I must admit to a little confusion here in thinking about the interaction between BGP's KeepAlive messages and the associated keepalive timers, and this new SendHold Timer. A common default setting of a BGP implementation is to pass Keepalive messages to a BGP peer every 60 seconds and if no such messages have been received in a 180 second interval the BGP connection is closed. The condition that the SendHold Timer appears to be the detection of an inability to send an outgoing message to a remote BGP peer, and the default value of this timer is 8 minutes. I can't help wondering that the remote end's session KeepAlive timer should surely have triggered by then and a keepalive notification sent to the local speaker and the session would've been closed. If these wedged TCP sessions are caused by an operator running a BGP session with a KeepAlives disabled, then I'm afraid that such foolhardy behaviour merits such broken responses as the proliferation of Zombie routes!

This leads to further speculation on the nature and intent of BGP KeepAlive messages. Why do BGP KeepAlive messages have no payload? In a bidirectional protocol where the desire is to only keep the session in an active state only when messages are capable of being passed in both directions, the conventional protocol mechanism is to use a pair of monotonically increasing tokens. Each speaker starts with their own token value and increments it after sending the token value to the remote end in its next KeepAlive message. When a speaker receives this token value it echoes this value back to the remote end, together with the incremented local token value. A blocked session will not be receiving updated token values, and this can be detected by the sender in the received KeepAlive payloads as the token value would not be updated.

This seems to me a lighter touch change to BGP, where the addition of a pair of token values to KeepAlive messages could be part of the initial BGP session capability negotiation. If both ends support this extended KeepAlive message format, then both ends have a continuous feedback signal that the remote end is receiving and processing their BGP messages.

But, as far as I can tell, we've not heard of a vulnerability that allows a hostile off-path third party to "jam" a specific route into a specific router or even set up a wedged TCP session between two BGP speakers. While the route beacons point to background level of stuck routes that become Zombies, it is still unclear that this is a widespread problem. Without data that points to a significant scale of zombie routes in the current routing system, and until we reach such an unhappy point where hostile third parties can trigger this zombie state for targeted routes, then I am having some challenges in describing this behaviour with the same level of concern as more effective off-path hostile attack on BGP, such as the remote injected TCP Reset attack. I might have a greater risk tolerance than others, but I mostly agree with Gert Doring's original description of the same issue, that these routing zombies fall into the general category of "curiosities".

Disclaimer

Author

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

www.potaroo.net