November 2019
Geoff Huston

My IETF 106

The 106th meeting of the IETF was in Singapore in November 2019. As usual for the IETF, there were many Working Group meetings, and this report is definitely not an attempt to cover all of these meetings or even anything close to that. Here I've been highly selective and picked out just the items that I found interesting from the sessions I attended.

IEPG

Some twenty-five years ago the activity called the *Internet Engineering and Planning Group*, or IEPG, had a role and a purpose and even an RFC (RFC 1690). These days it has become a small and selected overture of the IETF week's areas of current interest for those who arrive a day or so early and want a gentle easing into the week. Lately, its themes have centred around BGP and the DNS and the IEPG session for this latest IETF meeting was no different. (www.iepg.org).

On the BGP side, there was FORT, a new open-source RPKI validator. As far as diversity of platforms and implementations goes, it's always better to have diversity as compared to a monoculture based on a single implementation, so this is a good thing to see. I haven't tried FORT myself yet, but if it is as good as NLnet's Routinator, and it certainly seems to be there, this will be a very neat RPKI validation tool indeed.

The larger picture of BGP route origin security is looking better, with NTT's Job Sniders providing a summary report that showed uptake of networks filtering out announcements of route objects that were considered to be invalid in the context of the SIDR secure routing framework. RPKI and ROAs are relative newcomers to the inter-domain routing environment. Before then there was extensive use of various forms of Internet Routing Registries (IRRs). This use of IRRs also has a twenty-five-year history and over this time they have proliferated and accumulated a fair share of aged cruft. Recent efforts to improve the IRR space include a cross-reference to the RPKI to identify and remove cruft from the IRR environment. This is not quite as challenging as it may sound, in that the valid RPKI-signed ROA set can be used to filter out older contradicting IRR objects, but whatever the degree of difficulty needed to perform this automated filtering of IRR data, the results is that the IRRs appear to have been revived and once more can be a useful tool to filter out unintended and possibly deliberate forms of abuse of the Internet's routing system.

On the DNS side, DNSSEC validation failure is the topic of a study by Japan's JPRS researchers. They used both active probing and passive measurement in this study. They noted that DNSKEY queries increased when the DNSSEC information failed to validate at the recursive resolver, but for lower-level domains, this may not be so readily detectable. This appears to imply that if you want to monitor a signed domain name for validation integrity, then it seems that active monitoring is a more reliable approach.

Private namespaces continue to be an issue for the DNS. The actions in the IETF that resulted in a special use names registry were somewhat controversial at the time and it continues to carry through to today. An earlier proposal in the IETF for a space that readily identifies private-use namespaces under a common TLD of **.alt** did not get very far, and a subsequent proposal within the ICANN community to

support the case for the designation of **.internal** as a common private use TLD has not received widespread enthusiastic support so far. There is some role confusion over the IETF's role as the custodian of the RFC 6761 Special Use names registry and ICANN's role as the name policy community for generic TLDs there is, of course, the best way to resolve such areas of confusion in the DNS is to divert the conversation to the choice of the actual label to use! We obviously need a better name for such a generic TLD, and this presentation advocated the use of **.zz** for such a purpose. Whether it will be used at all is an unknown factor here, and the continuation of a broad variety of undelegated names for what appears to be private context continues largely unabated. I'm unconvinced that **.alt**, **.internal**, **.zz** or any other label would provide a useful resolution to what appears to be an ongoing issue in the DNS, but there is no doubt that as the beauty contest for the "right" label continues we may well fixate on the choice of a label value and forget about why we wanted it in the first place!

Finally, I presented on the handling of NXDOMAIN answers in the DNS. The observation is that end-user queries for a non-existent domain name are amplified within the host and network systems, and a single initial transaction of trying to resolve the IP address of a domain name is transformed to close to 3 queries on average at the authoritative server. The reasons for this amplification appear to lie in aspects of Dual Stack Happy Eyeballs behaviours in stack end systems, DNS re-query timers and DNSSEC validation time lags, UDP packet duplication, aberrant DNS load balancers. Underneath all of these is the prevalent attitude that its quite okay to hammer the DNS mercilessly. One of the weaknesses of the distributed functionality of the Internet is that we can create unintended outcomes where the intended benefits in one activity, here a desire for responsive systems that do not create user-visible delay, create imposed cost in another domain, namely aggressive timers and liberal re-transmit practices that create extraneous load in the DNS recursion environment.

Congestion Control Research Group

The most radical, and perhaps the most successful part of the Internet Protocol Suite has been the TCP transport protocol. It removed the need for hop-by-hop reliability and replaced it with a single mechanism to manage both data integrity and flow control that was intended to result in efficient network loading. TCP turned out to be perhaps the most critical part of the entire protocol environment. It has been around for some 45 years now and you might think we know all there is to know about how to make it work well. Turns out that's just not the case and work on managing adaptive flows across a network is still a topic of innovation and evolution.

Google's BBR protocol is being revised with BBRv2. My experience with BBRv1 has been very satisfying, but only in a purely selfish manner! BBRv1 can be very unfair. I've seen my BBRv1 sessions clear out then occupy large expanses of network bandwidth for itself, which is another way of saying that this protocol can be prone to starving simultaneous loss-based TCP of network resources. For widespread adoption in the Internet BBR needed to maintain its profile of low buffer impact, but at the same time behave more fairly to loss-based TCP sessions that make more extensive use of network buffers. Part of the refinements of BBR v2 is improved internal processing paths to reduce the CPU usage of BBR to match that of Reno, CUBIC and DCTCP. They have also adjusted auto-sizing and ACK processing. Part of the rationale for these changes to BBR is a shift in the way we use the Internet. The days of maximising throughput for large data sets over constrained paths are pretty much over, and today the Internet is far more about adaptive bitrate video, RPCs and web pulls. Such network use tends to be throttled by the app, not the network, which means that the congestion window within the transport session is not the limiting factor. What this implies is that when the session is not congestion window-limited, then ACK processing can be simplified enormously, implying lower processing overheads in such cases. BBRv2 also changes the Linux ACK behaviour removing a limitation on ACK generation, improving the integrity of the feedback signal of the receive window. BBRv2 will not exceed an explicit loss rate target and will take into account Explicit Congestion Notification (ECN) feedback from the network, where it exists. It is probably not the last word on congestion control in network sessions, but it illustrates that this remains an area of research and technology evolution.

Facebook described COPA, a congestion control algorithm that is also delay sensitive. The algorithm uses a 10-second sliding window to establish the minimum RTT and compares the current standing RTT against this minimum RTT. The higher the variance of this current RTT from the minimum RTT the smaller the congestion window, which, in turn, will throttle the sending rate. Earlier experience with delay-based congestion control algorithms (such as TCP Vegas from 1994) have found that this technique quickly starves the session, so COPA uses a so-called *competitive mode* to detect other buffer filling flows and reduces the sensitivity of the algorithm to downward pressure from these other flows.

In this family of delay-based congestion control algorithms is also LEDBAT. Work on a revision of LEDBAT was presented with LEDBAT++. The changes include a shift from one-way delay to simpler round trip time measurements, a multiplicative decrease, a slower slow start approach and smaller increase amounts.

The reason for all this renewed study of delay-based congestion control seems to lie in the effort to design an efficient high-speed congestion control protocol that minimises any dependency on deep buffers within the network.

V6 Operations

I am never sure what to make of these IPv6 sessions. Was the designed-by-committee IPv6 specification so broken in the first place that 25 years later we are still finding fundamental protocol problems and proposing yet more changes to the protocol? Or, are we just nit-picking at the specification because we haven't been told to down tools and just stop work? Or has something else changed?

Of the three choices, I suspect it's the latter, namely a change in the environment. IPv4 was designed in an era of mainframe computers and aspects of the protocol design reflect that environment. IPv6 is not all that different from IPv4 in many respects. It may have been designed in a time when desktop personal computers proliferated but mobility was yet to achieve today's absolute dominance of the environment. The mobile industry has had no time for the IETF's Mobile IPv4 or Mobile IPv6 protocols that attempted to maintain an IP address binding that survived dynamic attachment to various host networks. Mobile devices simply ignore this and instead use a dynamic association of IP addresses on an opportunistic basis. This dynamic association raises some issues in protocol behaviour. One of the issues is neighbour discovery, where a new host performs a router solicitation and configures a global unicast address, but this does not imply that the router has performed a comparable operation to put the host into its neighbour cache. It waits for incoming traffic to perform this operation. If the incoming traffic is part of a packet burst the burst is effectively discarded. Should this be fixed in the IPv6 specification? Probably!

There was a time in the refinement of the IPv6 specification when we told ourselves that the new protocol would have such a large address space that everything could be uniquely addressed in IPv6 irrespective of its connected status to the public Internet. We seem to have gone in an entirely different direction and the demise of scoped addresses, coupled with the entry of Unique Local Addresses (ULAs) poses a few questions that are still unresolved. If a manufacturer wants to pre-configure a local-only use IPv6 device than how can this be done? A solution proposed here was to preconfigure the device to only accept address configuration (SLAAC or DHCPv6) if the address is a ULA address, and require manual intervention to broaden the scope to a global unicast address for the device. Again, it's not easy to tell if this is a good response to a known problem or just more nit-picking at the protocol.

Application Behaviour Considering DNS

Over the past couple of years, the IETF has worked on two privacy initiatives in the context of stub-to-recursive DNS queries, namely DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH). These tools are intended to replace the platform-defined stub resolver code and cloak the open UDP query response protocol inside a session layer that provides integrity and encryption. In theory, DoH is a superset of DoT, as both use TLS as the channel platform and DoT places DNS queries and responses directly into this TLS stream with DoH providers a further wrapper of HTML headers.

In practice, there is a fork in this DNS road, as DoT appears to have been seen as a platform tool, replacing the host platform's UDP port 53 channel, where DoH is seen as an application module, and implementations have been built for Firefox, Chrome, CURL and OkHttp. It's this latter scenario which was the focus of this session.

Part of the issue here is the redirection of DNS queries, with applications opting to use non-local DNS resolvers. In response to various objections of a relatively invisible form of DNS query redirection, there has been exploration of the concept of so-called "canary domains" which are intended to be locally scoped privately defined names whose presence halts an otherwise automatic use of DoH by an application. The motivation here is to designate a 'safe' domain that can be locally defined and its definition is intended to trigger a DNS policy action. Purely as an example, if the domain name **it.is.ok.to.use.doh.com** resolves, then the application should switch to use DoH. In the context of the Firefox browser, the canary name is currently **use-application-dns.**net. If this name results in a negative response (NXDOMAIN or NO DATA) then Firefox will not automatically use DoH. There is some desire to generalise this approach and share a common canary domain name for all applications that want to check if the local infrastructure is unwilling to permit the automatic use of DoH.

There are some fascinating questions about this work, some of which appear to question the basic architecture and assumptions about the DNS as we understood it. Is the DNS one universal substrate where any recursive resolver can query about any defined name? Or are we defining customised spaces where only particular resolvers may be used to query for particular names? Here there is a possible inference that if this recursive resolver is not used, then the name will not resolve at all. Should DOH recursive resolvers deliberately add a policy, in the sense of not providing a response for some classes of domain names, or performing NXDOMAIN substitution for other names? How should such a policy be described and how could applications or end-users make use of it? From there we head into so-called "resolverless DNS" where the DNS is provisioned as a "push" service, so that applications are provided with answers to queries that were never asked in the first place! We also head into the area of speculation about assuming the presence of a client HTML application, where the application is capable of accepting more than simple **application/dns-message** HTML messages, allowing the application and the DoH resolver to conduct an exchange of metadata about the resolver's function. Obviously, such a meta-conversation is not an option in DoT.

There was this historical concept of the DNS as a 'universal' namespace. Indeed, this universality could in and of itself define the Internet, where the use of a common and consistent public namespace essentially defines a common domain of discourse that is the public Internet. We are heading in a different direction, though where this is heading and how quickly remains to be seen.

DPRIVE Working Group

It seems odd to say it, but I believe that it's unclear whether adding privacy constructs into the DNS has been beneficial or disastrous for the DNS as a whole. It's certainly been disruptive to the DNS architecture as we used to understand it. Whether these changes increase end-user privacy or merely provide largely meaningless obfuscation remains to be seen. So far, we've seen initial efforts to improve channel security with specifications of the stub to recursive encrypted channels (DoH, DoT and DoQ), as well as eliminating DNS query verbosity (Qmin).

These secure encrypted channels provide a mechanism for the application environment to pass *over the top* of the ISP-provided DNS resolvers and use resolvers of the application's choice. Rather than being essentially trapped into using the DNS name resolution environment provided by the local network infrastructure these encrypted channels can act as tunnels through this infrastructure. This broadened choice in DNS name resolution has exposed the observation that the DNS is not a homogenous environment. Some resolvers censor names, others perform substitution of NXDOMAIN responses, or perform substitution in responses. Some resolvers perform no query logging, while others not only log all queries, but then sell the profile of the queries and queriers. If the user expectation is privacy and

integrity, then even with a secure channel are there still points of weakness. Some recursive resolvers monetise query logs, using the IP address of the stub resolver as a key to the end user's identity. Not all recursive resolvers are the same and perhaps users would be well served if their queries for certain domain names were steered towards particular recursive resolvers.

This forms part of the thinking behind an approach of using the DNS to guide applications, where a DNS name can, with the use of a particular resource record, nominate a recursive resolver to use via an encrypted channel, and in the case of a TLS session using encrypted SNI in the TLS startup, another field within the same resource record can provide the public key to use with encryption of the SNI field to connect to this recursive resolver. All this is DNSSEC signed, although given that this is effectively a directive to the stub resolver this DNSSEC-signing makes sense only when the stub resolver is performing DNSSEC validation itself. If you wanted to fracture the DNS and create names that only resolve in certain contexts and not in others, I guess you would do exactly this, all in the name of *good DNS privacy* which seems more than a little disingenuous to me. On the other hand, this elevation of the DNS into application contexts raises the question of how much of the DNS you need to take with you. If the DNS already provides authoritative server information then why not provide access methods in this same area and ditch the entire concept of intermediation.

The question here is whether we want to preserve the current architecture of the DNS and just add some magic coating to prevent inspection and interception, or whether we are willing to contemplate the application level models of privacy in a larger model of content distribution? This should be an interesting discussion.

Routing Security and SIDR

This topic space has moved from the Routing Area to the Operations Area, reflecting a position that the basic protocol, BGPSEC, is complete and it is now largely a matter for deployment. However, this was a premature call in many ways, and what we are left with is some of the elements of a secure routing environment without the cohesion that would bring it all together into an effective secured environment.

There are not all that many tools that can provide assurance of the integrity of data. The common workhorse is the concept of the public/private key pair. This construct is commonly packaged using an X.509 public key certificate data structure. Each operator with a certification authority has the ability to generate subordinate certificate products, and those products are published in their local certificate repository. A network operator attempting to validate all digital signatures that are used in the Internet's routing table needs to poll all such certificate repositories all of the time to ensure that have a complete and up-to-date local cache of all certificates and all signed products. In the current BPSEC architecture this role is left to the network operator, and the tool is a polling tool. The originally proposed technology was a synchronization protocol called *rsync*. This appears to have been a poor choice as it does not scale well. One response is to deprecate *rsync* and instead use a slightly different tool called *RRDP*. To me this still seems like a poor fit to the underlying problem. If the basic problem is to perform a reliable flooding of information to all subscribed parties then instead of putting the onus on the client to collect all products from all sources the alternative approach is to unite both publishers and clients into a cooperative approach that reliably floods information from providers across all clients. Oddly enough this is exactly the problem that is addressed by a routing protocol, and one could readily image a system where each publisher and each client are nodes in a space spanned by a distance vector routing algorithm.

The current operational environment has concentrated on the validity of route origination and has left the issue of the integrity of route propagation, namely the AS Path, in the *too hard* basket. An earlier effort in this space, *soBGP*, used the concept of *path plausibility*, where each AS signed digital attestations that enumerated all of the ASs of its eBGP peers. This approach had the interesting property that it was not possible to create a synthetic AS path that contained an AS that signed all of its eBGP peers unless you also included one of these neighbours. The approach is incrementally deployable, with the property that the larger the number of participating AS's the harder it becomes to construct an artificial path that does

not correspond to an actual path within the inter-AS BGP space. The current ASPA takes this approach and adds an element of policy, where an adjacency is marked as *client-to-provider* or *provider-to-client*. The intended outcome of this additional marking is to detect and prevent those forms of routing leaks where a client re-announces routes learned from one provider to another. In the ASPA world a AS path that contained a sub-sequence of *provider-to-client* followed by *client-to-provider*. This seems like a useful approach for a secured routing environment, yet the rate of progress of this draft from concept to implementation is close to stalled.

Human Rights Protocol Considerations

There were also a couple of fascinating presentations from the Human Rights Protocol Considerations Research Group.

The first is a study of routing borders and gateways in response to the continuing recourse to national shutdowns of Internet access by various national regimes. The research were looking at the number of what they termed "Border AS's" where there appears to be an eBGP session where the AS's involved are located in different countries. Without regard to traffic volumes, they define a concept they term "chokepoint potential" for each country, where the figure relates to the number of Border AS's that operate some defined proportion of border-crossing eBGP paths for the country. Their studies point to an increasing level of chokepoint potential over time for a number of countries, where a smaller number of network operators control larger proportion of inter-border crossings.

The second is a consideration of security within 5G networks. If the Internet has attracted a reputation of excessive credulity in its basic design, the mobile world was little different. Surveillance devices, or false base stations, have been identified that track the ISMI numbers of mobile devices within range. The 5G network is hardening up against such attacks, including the use of encryption of the permanent identifier of the device, time limited temporary identifiers and tokens, secure radio redirections and active efforts to detect false base stations. The underlying message here: no network is inherently trustable, and devices need to operate on the assumption that third party eavesdropping and interception are always possible in any and every network.

Disclaimer

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

Author

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

www.potaroo.net