February 2022
Geoff Huston

DNS OARC 37

There was a meeting of DNS Operations and Research group in February, DNS-OARC 37. These are my notes from the presentations that I found to be of interest.

Zone File Bug Hunting

The DNS is deceptively simple. Simple, in that are few choices in how to configure the zone information about each DNS zone, and deceptively so in that is looks easy enough for humans to operate. Yet we see again and again that humans get it wrong, and when the DNS zonefile is misconfigured then the impact is global and because of caching the impact is long lived. We've seen with Azure, Facebook and Slack outages in recent years that while the original problem can be identified and rectified quickly, the distributed caching behaviour means that it takes many hours for the bad information to be flushed out of DNS caches.

Part of the issue here is that some DNS records hide quite impressive complexity. Wildcard records, where the exacting match algorithm required its own RFC (RFC 4592), CNAME records, where again further clarification was required (RFC 2181), and DNAME records all manage to generate confusion. But its not just records that require complex processing (such as NAPTR (RFC 2915), and S-NAPTR (RFC6408), and the various forms of Dynamic Delegation Discovery (RFC3401)), but even the supposedly simple records, such as the nameserver (NS) records, have been the cause of unintentional misuse and abuse. The issue of underlying complexity lies in non-determinism of the resolution process and the interactions between cached information and responses, and the complexity of references, aliases and rewrite rules.

DNS operators typically perform testing and monitoring of changes, but neither the testing nor monitoring are comprehensive, so mishaps can still occur, and as noted, mishaps in the DNS can be quite persistent due to caching. The larger the zone file the more complex the texting process.

The tool described in this presentation, 'Groot', reduces zone files to be tested into a smaller set of equivalence classes, and then performs a conventional set of DNS tests of delegation consistency, lame delegations, rewrite loops, missing glue and unresolved references on this reduced data set. The claim here is that this initial data reduction has a major impact on the test load for very large zone files.

Adaptive DNSSEC

Technology design often involves engineering trade-offs, and DNSSEC is no different. Signing a zone makes it far more challenging to misrepresent the contents of a zone if the client performs validation, but at the same time there is a performance overhead. Responses are larger, and may trigger truncation and re-query over TCP, which reduces the capacity of the server in terms of query throughput. Pre-computed signatures are generally more efficient to serve, but there are a number of assumptions in this model that may not be necessarily the case all the time. For example, a DNSSEC-query for a non-existent name in a zone assumes that the responder has access to the entire zone so that it can find the predecessor record in the zone in order to find the signed NSEC or NSEC3 record that spans the non-existent query name. You can drop this requirement by performing 'black lies' in NSEC responses (RFC 4470), but this

incurs the overheads of dynamic signing. On the other hand, this form of NSEC record prevents zone walking.

This presentation on "Adaptive DNSSEC" proposes that the recursive resolver uses an adaptive approach and uses DNSEC to an authoritative server under low query loads, but under high load, as would be experienced in a random name attack, switches to use DNS over TLS (AuthDoT) and explicitly not querying for DNSSEC signatures. To mitigate the TCP head-of-line blocking issues the approach is refined marginally to use DNS over QUIC (AuthDoQ).

This brings up an old point of clarification in the DNS that has evidently not been repeated often enough: data authentication is not the same as transport security!

Post Quantum DNSSEC

Long before we managed to build digital computers, Turing's Halting Problem essentially defined the limits of what was computable in 1936. Shor's Algorithm from 1994 has a similar role for quantum computers, specifically on the problem of prime factorisation. Quantum computers are still in their early days and it's not clear how long it will be before they will be used, but that has not stopped work progressing on looking for cryptographic algorithms that are more resistant to being broken by quantum computers. The US National Institute of Standards (NIST) is in the process of selecting one or more port-quantum algorithms through a competition-like process. In a report from July 2020, NIST released a status report on the second round evaluation of these candidate post-quantum algorithms that "summarizes the 26 second-round candidate algorithms and identifies the ones selected to move forward to the third round of the competition.[…] These finalists will be considered for standardization at the end of the third round." (https://www.nist.gov/publications/status-report-second-round-nist-post-quantum-cryptography-standardization-process). One of these second-round finalists is the FALCON-512 digital signature algorithm, and this presentation reported on the experience of testing this algorithm for use by DNSSEC in a PowerDNS platform.

FALCON-512 is faster to generate keypairs than RSA-2048 and slower than ECDSA P-256. The DNSSEC validation times for FALCON-512 are comparable to these other two algorithms. The issue with FALCON-512, as with most of these port-quantum algorithms, is that the key and signature sizes are large. A FALCON-512 signature is around 600 octets in size, and a public key ais 1,000 octets, which is similar to the sizes of RSA-2048 signatures and public keys. If a client wishes to avoid IP fragmentation, then it is going to rely on DNS truncation and fallback to TCP to get signed responses.

While we await the results of the third round of NIST's evaluation it's not possible to say what algorithm(s) we will need to support in this area but given FALCON-512 has size and performance properties not that dissimilar to RSA-2048, there are reasonable grounds to conclude that FALCON-512 would certainly be viable from a DNSSEC perspective.

Use of DANE/TLSA in MX Records

I think it would be accurate to say that DANE has not been a runaway success. In the browser world there is a stubborn persistence to use authentication based on the Web PKI. However, we were told that DANE was uses in the world of mail, helping to support various anti-spam measures by adding capability to authenticate a domain name. This presentation presents the results of a survey of the use of DANE records (TLSA records) for port 25 Mail Exchange (MX) records. The survey used zone data from zonefiles.io and performed a MX lookup on each name, stripping out the common central MX profilers including Google. Hotmail and Outlook. A DANE TLSA query was performed on the resultant 66M MX domain names. Of the 9M unique MX domain names, some 152K names had a TLSA record.

The most prevalent area with TLSA records are within the .NL (Netherlands) domain. The dominant TLSA usage form in the mail space is a self-signed X.509 certificate, with the fingerprint of the Subject Public Key Identifier (SPKI) and the SHA-256 hashsum of the certificate. The conclusion of this survey

is that the overall level of DANE/TLSA records is still marginal, within the single exception of the NL domain.

DNS Resolver Consistency

How do you ensure that the resolver code you are maintaining is working 'correctly'? Well, I guess you could read through the multitude of RFCs and attempt to cross-correlate the specifications in these documents against the behaviour of your code. Yes, that's a lot of work, and if you get down in the details of these myriad RFCs you will encounter areas where one RFC specification says 'white' while another says 'black'!

There is another way. Take an existing resolver implementation as a comparison and feed a large query set into the two resolvers and compare and contrast those queries and responses that differ. This is what Microsoft have done with their resolver implementation, comparing their resolver's behaviour against the Unbound resolver implementation.

It is interesting to look at the set of cases where behaviours differ. For example, when a response cannot fit into the USP payload the responder needs to truncate the answer section and set the TC=1 bit to indicate that truncation has occurred. Should a resolver just send back an empty answer section, just reporting back the query and the truncation indication? Or should the resolver include what it can into the answer section and truncate it? How should Glue records be handled? Should the resolver mark a response as truncated if all the glue records cannot be loaded into the response? How long should a resolver work on a query before giving up? Does it even give up? Should a resolver cache failure to resolve? For how long? If there is a stale cache entry, should you use it when resolution fails?

All these RFCs have been unable to tie down every aspect of DNS behaviours, and there is still much that is left to the implementor. Is it "better" for the DNS to have every resolver implantation precisely copy the behaviour of the other implementations? Or is there a legitimate role for variation in resolution behaviour, where the relevant standard specifications permit some level of implementation discretion?

Failure Behaviour in the DNS

Failure can expose a whole range of implementation behaviours that are not normally visible when everything is working as expected. When an application encounters an unexpected break in the communication flow how long should it wait before giving up and returning an error condition? While waiting for a response how many times should it resend the last packet? There is the view that it's better for the user experience (and better for avoiding becoming a DOS cannon) to fail quickly with minimal retries. And there is the view, more prevalent in earlier days, that patience and persistence in attempting to re-establish the session is often rewarded. In TCP most of this behaviour is governed by session timers, with the one area of the initial SYN exchange being where we see variations in behaviours. In DNS over UDP this is a relevant question. What should a client of a DNS recursive resolver do when the recursive resolver fails to answer after "a while"? And what should a DNS recursive resolver do when an authoritative server fails to answer after "a while"? Are the two cases much the same? Or should recursive resolvers use a different strategy? How persistent should they be in seeking a response? What are the impacts on the remainder of the DNS infrastructure when this happens?

When Facebook's authoritative servers failed to respond for six hours in October 2021, the queries seen on the .com/net nameservers increased from a 'normal' 7K queries per second (qps) to a peak of 900,000 qps, an increase of 128 times over the normal query load. Other experiments show a larger 1200 times increase in query traffic when a botnet domain is intentionally disrupted to return the SERVFAIL response code. A large increase in query load was observer at A and J root when the old key was published with the revocation bit, which abruptly disappeared when the revoked key was removed from the root zone.

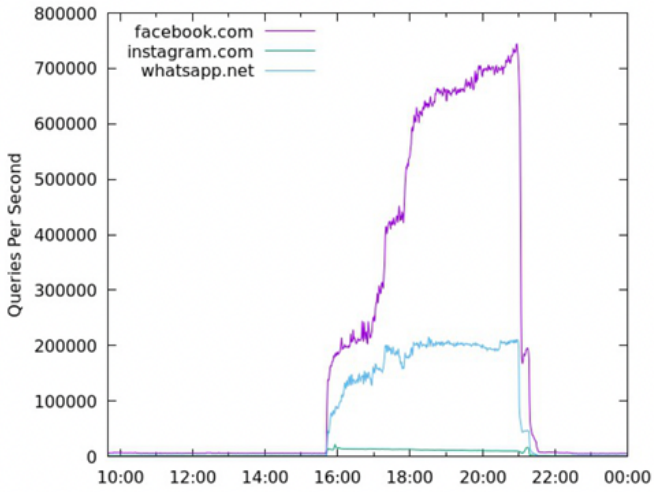

Figure 1 – Query Rate at com/net servers during Facebook Outage – From "Updating Requirements for Caching DNS Resolution Failures", Duane Wessels

DNS specifications have been progressively updated to provide guidance on how to handle the prolonged slience that follows a UDP query to an unresponsive server. RFC 8767 advises that "Attempts to refresh from non-responsive or otherwise failing authoritative nameservers are recommended to be done no more frequently than every 30 seconds.". However, this is perhaps a very coarse response and it leaves open the prospect of shutting down queries to a server in cases where the problem is context specific and related to the query, not necessarily the server.

There is a new draft, 'draft-dwmtwc-dnsop-caching-resolution-failures' in the DNSOP Working Group of the IETF. The document updates RFC 2308 by explicitly requiring the resolver to cache the non-responsive state of the server against the query profile. Further queries for this query should (well, in the IETF terminology it's a "MUST") not be performed until the cached information expires. It also provides some guidance as to what is unresponsive, and here it's a sequence of three queries. It is not as specific as to how 'unresponsive' is defined but suggests a local timeout of between 3 and 30 seconds. What about the timer to be applied to the cache entry? The document proposes a 5 second initial cache time, and an increasing cache time for subsequent resolution failures to a maximum of 300 seconds. Constraints should apply to querying the parent zone servers, and RFC 4697 describes such constraints, although is not as specific with suggested timers and retry limits.

More Data

Aspects of the DNS are quite opaque. In many cases this is related to the sensitive nature of the data and privacy considerations. In other cases, the data is just not readily available for researchers. ICANN Research is publishing two long time series data sets to assist researchers in exposing longer term behaviours of the DNS. The first data set is a daily pass across the top-level domain names (TLDs), collecting details of the TLD including the DNSSEC signing status and key profile, delegation details and similar. The second, the "DNS Census Core" is a larger set of zones that include the TLDs, ARPA delegations, various second level domains and other zones that are part of the share public DNS. This data set includes name server records, authority data and similar.

IP Fragmentation and the DNS

IP fragmentation has had a mixed history in the Internet. On the one hand it offers a very lightweight and flexible approach of IP adaptation to the characteristics of various network media. On the other hand, it poses a whole new set of security vulnerabilities. How can a receiver tell if an attacker has substituted its own fragment into the data stream? If the checksums match, then it can't. (https://blog.apnic.net/2019/07/12/its-time-to-consider-avoiding-ip-fragmentation-in-the-dns/). This can permit a class of attacks in the DNS, which can be most effective against unsigned domains.

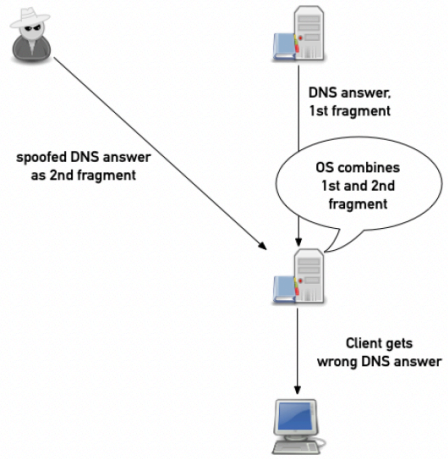

Figure 2 – IP Fragmentation Attack – From "IP Fragmentation and DNS Cache Poisoning", Carsten Strotmann

Recent efforts to measure the incidence of such cache-poisoning fragment attacks appears to be minimal, yet present. A response to this form of vulnerability is to use a EDNS UDP buffer size that would normally avoid fragmentation of DNS responses and use a packet filter to drop all DNS UDP fragments. TCP is less prone to fragmentation attacks, and session level encryption can protect the TCP payload again such efforts of content corruption, although they have their own incremental costs in terms of efficiency of DNS transactions and lack of universal support in DNS resolvers.

However, as with other forms of attack on DNS content I would still say that the best response is to use DNSSEC and sign the zone, and for resolvers, both stub resolvers and recursive resolvers, to validate DNS responses.

DNS OARC 37

The set of presentations from DNS OARC 37 can be found at https://indico.dns-oarc.net/event/42/timetable/#20220217.detailed. The recordings of the presentations will be posted to Youtube in the coming weeks at https://www.youtube.com/dns-oarc.

Disclaimer

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

Author

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

www.potaroo.net