Geoff Huston
November 2017

# Three DNS articles:

## 3. Helping Resolvers to help the DNS

In this final article of a trio that looks at today's "hot" topics in the DNS, I'd like to look at ways that recursive resolvers in the DNS can take some further steps that assist other parts of the DNS, notably the set of authoritative name servers, including root zone servers, to function more efficiently, and to mitigate some of the negative consequences if these authoritative name servers are exposed to damaging DOS attacks. If all of the 10 million or so active recursive resolvers in the Internet supported these mechanisms, the DNS would be a vastly different environment. If even just the top 7,000 or so resolvers that server some 95% of the Internet's DNS query load supported these mechanisms, it would still be enough to completely change the nature of the DNS. What are these steps that resolvers could take?

### RFC8192 – Aggressive NSEC Caching

Changing the way in which resolvers cache the information relating to the non-existence of names from DNSSEC-signed zones can have a major impact on the DNS and its resilience to attack.

The vulnerability that the DNS has in this space is a "random name" attack where the attack consists of posing an overwhelmingly large number of queries for names in the same zone. The attack is effective in the mode of a widely distributed attack, where each element of the co-opted zombie set generates just a few random name queries per second, which sits well below a conventionally detectable threshold. The attack becomes toxic at the point where to co-opted zombie set numbers in the millions or larger.

If the query name was constant, the caches in the DNS' recursive resolver population would be able to answer the queries, and therefore protect the authoritative name server. The use of random name strings defeats these conventional caches and the recursive resolver will pass the cache miss queries to the authoritative name server, and all the names served by this authoritative name server will start to fade away as the recursive resolvers time out their local cache. (See http://bit.ly/2jUnMRv for a further description of this approach in the context of the October 2016 Mirai attack.)

This is where DNSSEC can help. If the zone is DNSSEC-signed, and the resolver both requests DNSSEC credentials, and performs DNSSEC validation, then the response received from the authoritative name server for a non-existent domain name contains more information than was strictly requested, to facilitate the resolver to perform validation of this response. The NSEC response describes a range of labels that are not defined in the zone, and does so by listing in a signed response the two labels that are contained in the zone that minimally span the non-existent label. If this label range is cached by the recursive resolver, rather than just the queried label, then the same NSEC response can be used to respond to subsequent queries that relate to any label that sits within the NSEC-defined span of labels within this zone.

When handling queries related to a random name attack, a recursive resolver will rapidly "learn" the entire contents of the target zone, and at that point will have no further need to consult the authoritative name server until the cached entries expire.

To understand just how dramatic the change could be for the DNS if all names were DNSSEC signed and all recursive resolvers performed NSEC caching, it has been observed that some 70% of all queries to the root name servers relate to non-existent top level names. It all recursive resolvers by default performed DNSSEC validation and aggressive NSEC caching of just the root zone, we could drop the query rate to the root servers by most of this 70%, which is a dramatic change. A broader study of queries to the root observed that some 94% of all queries elicited NXDOMAIN responses. DNSSEC zone-signing and NSEC caching has the potential to absorb a large proportion of this query traffic at the recursive resolver.

## Cache Times

The query traffic that is passed to authoritative name servers from recursive resolvers consists of queries relating to previously uncached names, and names that have expired from the cache. Another method that can drop the query rate on authoritative name servers is to increase the cache lifetime settings for the zone.

The amount of time a response can sit in a recursive resolver's cache and be used for further responses without reference to the zone's authoritative servers is defines as the Time To Live (TTL). The DNS specification itself does not define a single TTL, nor should recursive resolvers invent their own TTL. A recursive resolver is meant to follow the directives given by the zone's publisher and cache records to the period specified by the zone.

Up until the publication of RFC2308 in March 1998, the TTL field of the Start of Authority (SOA) record in a zone file was overloaded with three somewhat different meanings: the minimum value of all TTL records of all Resource Records (RRs), the default TTL value for all RRs that do not explicitly set a TTL, and the TTL of cached negative responses. RFC2308 deprecated the minimum TTL concept, so this is no longer relevant. The second interpretation, that of the default TTL for RRS where no TTL is given was reassigned to the $TTL directive in the zone. This leaves the TTL in the SOA to be the negative response TTL.

There are many reasons why short TTLs are used, and these relate to the ability of changes to a zone to be promulgated through the DNS quickly. If the $TTL and SOA TTL values were both set to, say, 604800 (one week), then any changes to the zone would take on average 3 ½ days to be reflected in a recursive resolver that contains the information in its cache, and as long as a week if the cached RR had been stashed into the cache just prior to the change.

On the other hand, the longer the cache time the more effective the recursive resolver is in intercepting queries and using its cache to generate responses. This is a consideration for caching of NSEC records and the desire to push random name query attacks back to the recursive resolvers. There is no magic answer here, but longer cache times obviously provide better caching outcomes, at the cost of timeliness of promulgation of changes to the zone. In any case, try not to make the cache TTL longer than the signature validity period in any case for DNSSEC-signed zones.

## Serve Stale

DNS attacks on the authoritative DNS servers for a zone are intended to 'effectively remove the RR for the network by preventing recursive resolvers from resolving the name. DNS resolvers do not synchronize their actions between themselves, so the probability that a resolver's cached information will expire at some given time is equally distributed across the RR's TTL interval. This means that under a sustained attack that prevents all of a zone's authoritative name servers from responding at all to queries, the RR's within the zone will gradually fade away. Once a recursive resolver expires the RR

information from its cache at the end of its defined TTL it will be unable to respond to further queries until it can once more reach the zone's authoritative name servers.

The observation here is that the richness of expression that allows a zone to define the interaction between the primary authoritative source of information and its secondary servers is missing from the RR records within the zone. There are no comparable defined timers to define a **Refresh** interval (the intended number of seconds between successive refresh queries from a secondary to the primary), the **Retry** interval (the number of seconds between refresh queries when the authoritative server is unresponsive) and the **Expiry** interval (the number of seconds since the last zone refresh that defines when the secondary server should stop serving this zone. This, together with the SOA record, which is a "signature" of whether a zone has changed, allows the zone administrator to define the chosen characteristics of a zone. If the zone changes rarely then availability might be more important than "freshness" and longer values for these three timers might make sense. For zones where 'freshness' is of primary importance, then shorter timer values is sensible. RR records do not have such ability to control caching behaviour. There is a single timer that defines the time a RR is allowed to be served from a local cache.

What if we were to define a similar way to define the preferred behaviour of caching recursive resolvers? We might want to look at a refresh time, which like the current RR TTL, defines the time that a RR can reside in a cache without further reference to an authoritative server. But then we might want to define a longer expiry time, which allows a recursive resolver to serve otherwise valid data from its cache without a refresh. And perhaps a retry time to define the interval between refresh query attempts would also help if we went down this path.

A similar approach has been proposed as "DNS Serve Stale" (draft-tale-dnsop-serve-stale: http://bit.ly/2A9xZjl). This is a more nuanced approach that advocates a subtle modification of the definition of the RR's TTL value:

> TTL a 32 bit unsigned integer number of seconds in the range 0-2147483647 that specifies the time interval that the resource record MAY be cached before the source of the information MUST again be consulted. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached. Values with the high order bit set SHOULD be capped at no more than 2147483647. If the authority for the data is unavailable when attempting to refresh the data past the given interval, the record MAY be used as though it has a remaining TTL of 1 second.

The intent here is to mitigate some of potential outcomes of a DOS attack mounted against the authoritative servers of a zone: even if the authoritative name servers of a zone are unresponsive, recursive resolvers would still be able to serve RRs, assuming of course that they exist in resolvers' local caches.

For DNSSEC-signed RRs the maximum cache lifetime has an upper bound of the expiration time of the DNSSEC signatures used to validate the cached entry. It makes no sense to serve RRs that cannot be validated.

## DNS Hammer

The internet draft of the Hammer behaviour probably should be awarded a prize for one of the most tortured acronyms of 2017. "Hammer" evidently stands for "Highly Automated Method for Maintaining Expiring Records" (http://bit.ly/2i8RbHd). This draft describes the option for a recursive resolver to perform "pre-fetch" of an expiring cached item, and to do so prior to the item's scheduled cache expire time.

This behaviour does not impact the DNS query load on the authoritative name servers, but the deliberate operation of a loaded cache is intended to ensure as high a cache hit rate as possible for extended periods of time, improving the DNS resolution performance of its DNS client population.

Hammer does not extend the life of a cached RR, but it does offer a way for a recursive resolver to keep the most popularly queried RRs in the cache, and not unduly penalise the random user who happens to issue the query at just the wrong time and encounters the cache miss.

I wonder if DNSSEC-signed zones can make this Hammer operation more efficient. For the synchronisation of secondary servers to the authoritative server, a zone's SOA record is a convenient flag to indicate a change to the zone, so that secondary servers can track the SOA record and look for a change in the index value as a signal to refresh the local copy. While RRs have no concept of a SOA index, signed zones might be able to use the RRSIG signature record. If there is no change to the RR value the RRSIG would be unaltered, and it there was a change then the retrieved RR would have to be DNSSEC validated in any case.

## Conclusions

It is possible to change the behaviour of the DNS, and do so in a manner that places more overall functionality into the responsibility of DNS recursive resolvers. This way we can exploit the large distributed population of these devices and use them more effectively, in order to significantly improve the resilience and resolution performance of the DNS. The changes contemplated here are all relatively minor in nature, yet when combined in recursive resolvers would make a significant difference to the operation of the DNS as a whole.

## 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*

## Disclaimer

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