March 2018
Geoff Huston

DNS OARC 28

March has seen the first of the DNS Operations, Analysis, and Research Center (OARC) workshops for the year, where two days where too much DNS is just not enough! These workshops are a concentrated two days of presentations and discussions that focus exclusively in the current state of the DNS. Here are my impressions of the meeting.

DNS as a Load Balancer

When you have a set of replicated content servers spread across the Internet how do you share the load so that each user is directed to a server that can offer the user the best service? You could put the services behind the same IP address and leave it to the routing system to select the closest (in routing terms) server, but if the closest server is under intense load routing alone does not correct the problem. Facebook use a slightly different approach, where each service point is uniquely addressed. They use a rather old technique that was first widely promoted by Netscape many years ago, where they use the DNS to steer the user to a close service point that has available capacity.

To a first level of approximation this apparently woks acceptably well, but it is a coarse tool. Caching of the DNS response mean that it's not easy to steer this service traffic with any high degree of precision and timeliness, and the widespread use of non-local open DNS resolvers mans that the user may not be located 'close' to their DNS resolver so the location information associated with a resolver's IP address may be misleading in such cases. EDNS(0) client subnet signalling could assist here, but it only does so at the expense of a certain level of privacy leakage.

Don't Give em Rope!

Robert Edmonds of Fastly noted that consumer home gateway servers distributed by the larger home ISP retailers do not provide a configuration option to provide the IP address(es) of a chosen DNS resolver. The implication is that the home gateway is configured with the ISP's own DNS resolver, and by default it will pass the consumer's DNS queries to the ISP's selected DNS resolver. It's easy to see why an ISP would find this attractive. It reduces the number of variable factors in the user environment, and thus reduces the number of variables that may cause the user to ask the ISP's helpline for assistance (a service that is generally quite expensive for the ISP to provide). Giving the user too much rope allows the user to get into all sorts of trouble and that involves often costly remediation efforts. By reducing the number of variables in the user's environment, the possibility of the user configuring themselves into a broken situation is reduced, so the principle of keeping the rope as short as possible is often used in these devices.

Of course, there is also the prospect that the DNS query feed provides a useful insight into the online activities of the consumer base, and in a world of declining margins for access ISPs the temptation to monetise this particular data flow may have already been overwhelming. This position on whether the gateway acts as a DNS interceptor and prevents local devices from passing DNS packets through the gateway to other DNS resolvers also appears to be a mixed picture. Some evidently work as interceptors while others will pass through the DNS queries.

The picture gets a little more involved with gateways that also provide a NAT function. Trailing fragments of a large UDP DNS response have no port field, so the gateway either has to perform full packet reassembly to get the IP address translation correct for the trailing fragments, or it simply drops the trailing fragments and leaves it to the home device to work around the damage! The more robust approach from the perspective of the CPE device is to place a small DNS engine in the CPE itself and forward all local queries to an ISP-configured resolver. That way the CPE itself accepts the external DNS responses and passes them into the local network without the need to perform firewall magic on fragmented DNS responses.

NSEC Caching Measurements

DNSSEC is not just about getting a trustable answer from the DNS. The mechanism used in DNSSEC to provide authenticated denial of existence provides a whole lot more than just the nonexistence of a name in a zone, as the DNSSEC response provides an entire range of names in the zone that does not exist. Ordinarily this would not really matter much either way, as long as you assume that the DNS is there to provide answers to names that exist in the DNS. But there are folk who abuse the DNS in all kinds of ways, and one form of attack is a "random name attack". If you train a compliant bot army to issue DNS queries for randomly generated strings in a delegated zone then because every name is a unique name its non-existence will not have been cached in a recursive resolver, and all of these queries will find their way to the authoritative name server(s).

A sustained attack will not only cause an impact of the availability of the attacked name, but also has the potential to negatively impact on the cache efficiency of the recursive resolvers as well, as a resolver's cache could end up being fully populated by these non-existent names. If the attacked zone is a DNSSEC-signed zone and recursive resolvers performed caching of NSEC records (as described in RFC8145) then we could well have an effective answer to this particular form of random name attack.

This mechanism has been implemented in the Bind 9.12 DNS resolver, and CZNIC's Petr Špaček related his experience in testing this with a replay of anonymised query traffic gathered from a public resolver, replayed through a test trig that used NSEC caching. in this query set 14% of the queries had NXDOMAIN responses. Oddly enough, he noticed a very slight increase in the time taken by the recursive resolver to respond to queries, of the order of a small number of milliseconds, which would not be expected to be visible to users. The number of packets sent to the authoritative name servers, and the total bandwidth of these packets dropped by 1-2% in packet count and 2-3% in bandwidth.

The study then used the same NSEC caching mechanism to test a random name attack. For small zones, the attack was completely absorbed by the recursive name server within 2 seconds. A larger zone, with 14,000 labels, took slightly longer at 14 seconds to have a random name attack entirely absorbed by the recursive name server. The larger the zone the smaller the span of each NSEC record and the longer it takes for the recursive resolver to completely 'learn' the contents of the one via the returned NSEC records. This is of course entirely expected. In sparsely occupied zones the "negative" space is far denser than the "occupied" space, so each NSEC span encompasses a significant proportion of the total random name space. The larger the population of defined names in a zone, the smaller the average span of NSEC records, and the longer the time it takes for a resolver to populate its local cache based on processing random names.

DNSSEC is Stranger than You Thought

DNSSEC is based on a simple concept of overlocking keys that match domain name delegation. A zone's key is signed by the zone's parent key, and this is recursively applied all the way to the root zone. Trust is therefore a process of validating this chain of interlocking signatures all the way to the published root KSK key.

What happens when I want to change my key? Well, you need to tell your zone parent about the new key, add the new key into the DNSKEY record for the zone. The next step is to sign the DNSKEY records with the incoming key, then retire the old key and let your parent know the old key can be removed from their zone as well. We thought we understood the steps and documented the procedure in RFC 6781.

Roy Arends reported on the curious situation where a a number of signed zones were in the process of performing this KSK roll, and during their process the zones became unverifiable, even though the process was being followed assiduously.

Why?

Cryptographic algorithms are not perfect. They are normally just good enough for encrypting to be feasible with current capability, but too hard to decrypt with a similar technology capability. As technology improves, then best practices suggests that algorithms be updated, and DNSSEC is no exception. RFC4509 updates the DNSSEC specification in RFC4035, by proposing the use of SHA-256 digest algorithms in preference to SHA-1. The document includes the advice:

```
"Validator implementations SHOULD ignore DS RRs containing SHA-1 digests if DS
RRs with SHA-256 digests are present in the DS RRset."
```

It also includes the pragmatic advice that:

```
"Because zone administrators cannot control the deployment speed of support for
SHA-256 in validators that may be referencing any of their zones, zone operators
should consider deploying both SHA-1 and SHA-256 based DS records.  This should
be done for every DNSKEY for which DS records are being generated."
```

It appears that there is strong advice in RFC4509 to use SHA-256 and generate DS records that contain both SHA-1 and SHA-256 digests.

Let's glance again at that KSK roll process described in RFC6761. When the new DS record containing the SHA-256 digest of the new KSK key is incorporated into the parent zone, then validating resolvers will immediately ignore the old DS recorded prefer the new one. But at this stage the old key is still being used to sign the DNSKEY record, and the interlocking chain of keys is broken. The zone is now dark for DNSSEC-validating resolvers.

The missing advice on how to roll the KSK is simple: Do not roll the KSK and the DS digest type at the same time. Either roll the KSK or roll the DS digest type, but not both at once.

More generally, don't take everything you read in an RFC as the absolute and complete truth. There is always room for as-yet undiscovered errors and omissions!

In DNSSEC My Failure can be Your Problem

DNSSEC is intended to allow the client of the DNS to determine for themselves if they can trust the answer that they are provided from the DNS. Within a DNSSEC-validating resolver, then if the validation process fails then the resolver would withhold the response and pass back a ServFail response code. This is a robust response when there is some malicious attempt to pass off a faked DNS response, but it seems that a more unfortunate outcome is that many of the validation failures that have been seen are the result of mishandling of DNSSEC keys, particularly during the process of key roll.

As Comcast's Joseph Crowe noted, while it would normally be considered to be the problem of the zone administrator to find and fix their problem, there are some zones that are just too big to fail. And for operators of large scale DNS resolvers, then the responsibility for fixing the 'problem' becomes a shared responsibility.

Out of this came RFC 7646 and the concept of "Negative Trust Anchors". Negative Trust Anchors are local configuration element that allows the local resolver to return a response that clears the AD bit to indicate that validation was not performed. Automating this process is tempting, but at the same time automating a process that has the potential to mask an attack on the DNS could well turn out to be an ill-advised. So for many operators of large validating resolver services, either they allow key mishaps to pull the zone offline, or operate a mitigation service that necessarily appears to include some degree of manual oversight.

Steering the DNS between the IP Icebergs

The IP network has its problems. By "problems" I'm referring to issues at the level of pushing an IP packet into the network at one point and expecting to see it emerge at the intended exit point. Some of these problems emerge when the packet is large. IP has a relatively unique architecture of being able to 'fragment' a large packet into multiple smaller parts. This fragmentation is an IP level function, so the IP header is preserved across all the fragments, with fragmentation control fields to aid the remote end to reassemble the original IP packet. When sending fragmented DNS packets, it appears that many parts of the network appear to drop fragmented IP packets, particularly with IPv6. A previous measurement exercise measured a loss rate of some 38%, which is by any rational metric, a horrendous loss rate.

The DNS tries to steer around this, and related problems in transmission, by management of the EDNS(0) UDP buffer size. If the query elicits no answer from the server the local resolver may not just give up, but may issue a new query with a smaller EDNS(0) UDP buffer size. The intent is to see whether a smaller response, in the form of a truncated response, can make it through, or whether the server is truly offline If the client receives the truncated response then it is expected that it opens a new connection to the server in TCP, and the query and response is then repeated within this streaming protocol. All this takes time, as "receives no answer" is implemented as "waited for a while and nothing came back". The Internet is impatient and waiting at the protocol level is an anathema.

There was an interesting idea published as an Internet Draft back in September 2017 by Davey Song (https://tools.ietf.org/html/draft-song-atr-large-resp-00). It has a certain delightful simplicity: if a DNS server sends a fragmented UDP response to a client it then waits for a small period (10ms is suggested) and reflects back the query as a response, but with the truncated bit set. It's a shortcut to TCP without the wait. If the client can accept a fragmented UDP packet set, then the trailing truncated response will be ignored. If the fragments are dropped then the truncated response will likely be delivered, prompting the client to perform an immediate switch to use TCP for this query.

The results Joao Damas and I presented at the OARC meeting were encouraging. The very high fragmented UDP loss rate was reduced by over 10% when using ATR in both IPv4 and IPv6. I'll write more about this ATR test in its own article, but it shows that there are still good ideas out there that can improve the speed and reliability of the DNS.

Packing it In

The basic use of the DNS is that you get to ask one question and you get one answer. So, if you are a dual stack client and you want the IP address of some name, and you are perfectly capable of using either IP protocol then you probably ask two queries; namely the A and the AAAA resource records. If speeding up the DNS is the objective, then this is tedious and inefficient.

One approach, explored by Kazunori Fujiwara of JPRS, is to add an NSEC record into the response, so that DNSSEC-aware resolvers are able to start populating their cache with both names that exist and the scope of undefined names in a zone with just the queries to defined names. He then tested this approach with the common resolver implementations and found that some implementations already accept additional NSEC records in a response and use them to populate the local cache. Some implementations also accept A and AAAA records and store them in the local cache.

Conclusions

The DNS is certainly not an ossified protocol. It remains a basic protocol that underpins the operation of the Internet, and the efforts to make it more efficient, more resilient and more secure continue. And much of that effort is reported in these DNS OARC meetings.

The meeting materials are online at: https://indico.dns-oarc.net/event/28/timetable/#20180308.detailed

Disclaimer

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