February 2021
Geoff Huston

DNS OARC 34

It's an interesting topic of speculation to think about what form of network architecture would we be using if we were start afresh using today's world of scalable content and service distribution as the starting point. Like the "clean slate" discussions of over a decade ago, if we were to think about today's world without inherent assumptions based on unicast models of networks largely derived from the telephony service model, and we were to think about the network architecture in massively replicated service terms that are more like the old school publication or retail worlds, we probably would not have come up with a network architecture based on unicast destination endpoint address-based packet forwarding. Maybe such an Internet would be based on some form of what we've come to call *name-based networking*. We would probably start such a design exercise with the functions that today are found largely in the functions that are currently embedded in the DNS. What are these functions? What's going on in the DNS these days? One way to understand the current topics of focus in the DNS is to tune in to the regular meetings of the DNS Operations and Research community. OARC 34 was held in the first week of February and here are some items from that meeting that interested me. (https://indico.dns-oarc.net/event/37/timetable/#20210204.detailed)

DNS Flag Day 2020

Efforts to coordinate a common action from DNS software vendors has continued with the DNS Flag Day 2020, a program intended to stop the use of fragmented UDP in the DNS. By default, the DNS is a UDP-based transport and UDP can be incredibly fast and efficient. But there is a trade-off and the open DNS UDP payload is prone to all kinds of snooping and manipulation. In response, the DNS is in a slow but steady path to adopt DNSSEC signing, where digital signatures in the DNS are intended to protect the integrity of DNS responses for the client. (Another part of the response is to reposition the DNS to use encrypted channels, but that's a different story.)

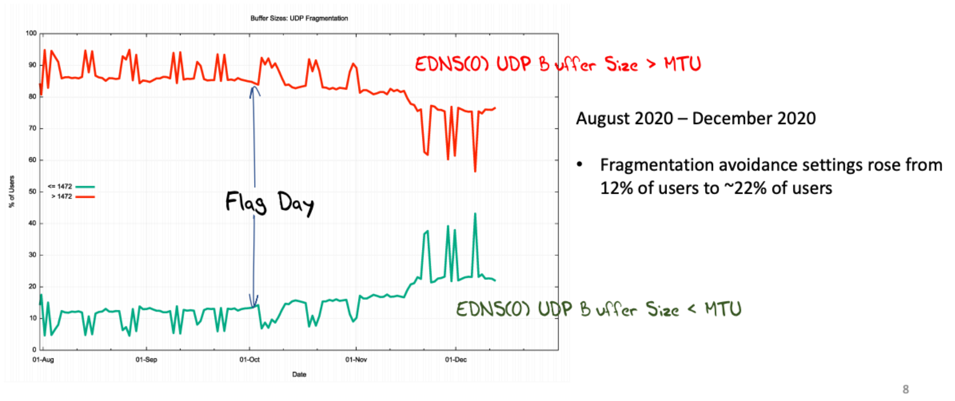
The problem with DNSSEC is that digital signatures tend to get large, and the DNS used a transport strategy originally devised in 1980s where IPv4 hosts could only offer an assurance that they would receive a packet of no more than 576 octets.

That limitation still applies today, where a standards compliant IPv4 host need only accept incoming IP packets of up to 576 octets in size, fragmented or otherwise. Pragmatically, most hosts these days are capably of accepting incoming IP packets up to 1,500 octets in size, a clear legacy of the ubiquity of the original Ethernet specification.

The DNS specification adopted a maximum UDP payload of 512 octets, which was consistent with this overall approach. The implication of this decision was that any form of additional payload content in the DNS, including DNSSEC digital signatures, would require switching to TCP, which would unacceptably slow and inefficient if we all started doing DNSSEC signing and validation. The response to this was a "tweak" mechanism for extending the DNS by including in the query a field that described the maximum size of a UDP response that the DNS client could accept. This allowed a DNS server (recursive resolver or authoritative server) to send back a UDP response whose size was up to the query-specified size before the server would signal the need to shifting to TCP by setting the truncation bit in the response. For DNS payload sizes over 1,472 octets (or 1,452 octets in IPv6) this implied that the DNS response would normally use fragmented UDP. Problem solved!

Well, no.

Because IP fragmentation presents its own security issues and many networks deliberately drop packet fragments as part of their traffic security profile. APNIC Labs measured an average drop rate for fragmented UDP between authoritative servers and recursive resolvers that would impact some 15% of the entire user base. It’s more challenging to gather broad-based measurements at the edge of the network, but there are grounds to believe that this fragmented UDP drop rate is far higher for packets passing from recursive to stub resolvers. That's a large enough number to say that UDP fragmentation is just not working for the DNS. However, we observed that many recursive resolvers use a default UDP buffer size of 4,096 octets in their queries. This occurs in some 80% to 95% of cases in the study.



August 2020 – December 2020

- Fragmentation avoidance settings rose from 12% of users to ~22% of users

Figure 1 – EDNS(0) Buffer Size Distribution
https://indico.dns-oarc.net/event/37/contributions/806/attachments/782/1366/2021-02-04-dns-flag.pdf

The intention of the DNS Flag Day 2020 was to stop using large buffer size values in DNS queries and bring the default buffer size value down to 1,232 octets, and thereby attempt to avoid UDP fragmentation.

Did it work?

Well not quite as well as intended. By December the use of large UDP buffer sizes dropped to 75% of cases. However, there was a rise in the use of both 1,232 and 1,400 octets in its place. Does one size setting match all cases for DNS use of UDP? The proposed value of 1,232 octets is a conservative value with a high assurance of fragmentation avoidance, but early onset of TCP extracts a marginal cost in terms of efficiency and speed of resolution. Maybe we could improve on this by tailoring the value to suit the context of the DNS query/response transaction? The measurements in this study suggest that in the “interior” of the Internet between recursive resolvers and authoritative servers has an effective packet MTU of 1,500 octets. Perhaps we should use a setting of 1,472 octets for IPv4 and 1,452 for IPv6 for the interior of the DNS for transactions between recursive resolvers and authoritative servers.

However, shifting to TCP has a high price to pay. There is client and server state to maintain and TCP has its own issues with robustness, with an observed failure rate of some 2% of cases. I found the ATR approach to be an interesting response (https://tools.ietf.org/html/draft-song-atr-large-resp-03). In ATR the server will attempt to provide a UDP response of size up to the query-specified buffer size, and if this causes fragmentation of the outgoing UDP packet the server will send an additional response of a truncated response with an empty answer section after a short delay (10ms is suggested). In the fragmented UDP response is dropped, there is a high prob ability that the small truncated UDP response will be received, causing an immediate re-query using TCP, avoiding client-side timeouts and continued UDP re-queries. (My thoughts on this ATR approach, together with some measurement data, can be found at https://www.potaroo.net/ispcol/2018-04/atr.html.)

DGAs and Viruses

Like their biological counterparts, computer viruses live in a constantly shifting world where the virus authors try to keep a step ahead of the countermeasures that are deployed in host platforms and networks. In November 2008, Conficker A, the first of five variants of the Conficker family of malware, rapidly began infecting computers which had failed to install a Microsoft patch released just weeks earlier. Conficker was malware intended to create a massive Botnet. Conficker levered the observation that the DNS served a useful function as a ubiquitous information channel that could be exploited for command and control and took it a step further by using up to 50,000 pseudo-randomly generated domains (using Domain Generation Algorithms, or DGAs) per day to keep in contact with the Conficker command and control system.

At the time Conficker was unleashed on the network the basic exploit in the Microsoft platform was already addressed, but this did not appear to curb the virulence of the malware. More than a dozen years later there is still visible Conficker activity. Far sight's Eric Ziegast reported on the ongoing efforts to track usage of the Conficker's dynamically generated domain names in 2021 using a random name generator and the DITL collection of queries to the root name servers.

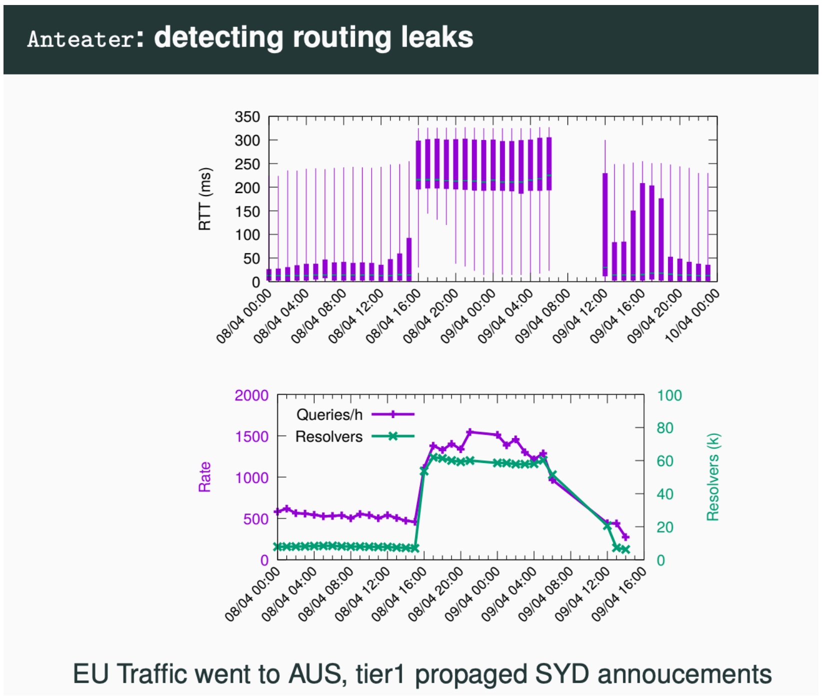
I find it depressing to see that Conficker is still out there. I find it even more depressing to think that the entire premise of the Internet of Trash is that we can flood our world with unmanaged devices and not have them turn into a billion-strong zombie attack army. Given that we can't even eradicate a well understood and exhaustively dissected Windows 95 virus from the network, then it seems like completely wishful thinking that we are getting any better at the job of managing the risks associated with malware and viruses.

TCP Telemetry in the DNS

Despite the current concerns about DNS privacy, it has to be admitted that the DNS is quite opaque. DNS queries contain the query and nothing more. Who is asking? Why are they asking? When did they ask? DNS responses are similarly terse. Who provided this response? If it was from a cache, then how stale is the cached entry? At APNIC Labs we've looked at measurement frameworks that can provide some answers to these questions, but they rely on seeding the DNS with queries where the query name itself embeds some essential information about the location seed point and the time of the seeding. But the distribution platform is limited in a number of ways, and it is not an effective general purpose measurement tool. The question remains as to how one can use existing query and response data from recursive resolvers and/or authoritative servers to answer these questions.

The motivation behind the measurement is pretty simple. Scaling the DNS has involved replicating the information and serving it from multiple locations. We used to achieve this in the authoritative space with constellations of secondary name servers, but there was no immediate solution to load distribution for recursive resolvers. We've largely shifted the DNS into anycast service formats where we use the routing system to direct clients to the "closest" service instance. We can scale up DNS services for both recursive resolvers and authoritative servers by increasing the number of distinct service platforms that share a common IP service address set, and use the rouging system too spread the local across the servers. The lingering question remains: how effective is anycast? BGP path selection is not based on lowest latency, nor on highest network path capacity. When we use anycast we are implicitly hoping that the lowest BGP path metric exposes a path to a server which offers the lowest delay and highest capacity.

This presentation from Giovanne Moura of SIDN Labs reminded us of an old but still effective measurement approach that uses analysis of TCP connections. The initial three-way TCP handshake provides a good estimate of the round-trip time between the client and the server. The client can use the elapsed time between the initial sent SYN and the receipt of the matching SYN/ACK and the server can measure the elapsed time between the sent SYN/ACK and the ensuing ACK. The presentation presented some measurements to confirm that TCP is used widely enough to provide a broadly useful measurement base, and the measurements in TCP correlate well with parallel measurements using instrumented probes. As a measurement tool it can be used to measure the effectiveness of anycast configurations, identifying collections of clients who are poorly served by the existing anycast clouds, or identifying when anomalies occur in routing that cause traffic diversion in anycast (Figure 2).


Anteater: detecting routing leaks

EU Traffic went to AUS, tier1 propaged SYD annoucements

Figure 2 – Detecting anycast routing leaks using TCP handshake timing data
https://indico.dns-oarc.net/event/37/contributions/812/attachments/790/1391/Moura-DNSRTT-OARC34.pdf

DoH Trials

I'm not sure any of the proponents of DNS over HTTPS (DoH) gave the considerations of fielding DoH servers all that much thought. After all, it's not all that different from the deployment of any other form of HTTPS server environment. The server has to set up TCP connections, negotiate a TLS session, connect to an HTTP application server. Almost the only differences is that the HTTP application passes incoming queries to a DNS engine and collects and delivers responses. One would expect that the only change is a substitution of some form of content retrieval engine with a DNS engine.

Perhaps it's more subtle than this. These days the expertise in operating large scale content servers is migrating over to the CDN operators, and DNS infrastructure operators have been focussed on tuning high capacity UDP transaction engines As Charter Communication's Jason Weil and Todd Medbury pointed out, there is certainly a mind shift to support DoH in terms of operations. They encountered some subtle factors, such as the key size and algorithm used for the TLS session. Given that the DNS transactions are often small the overheads of TLS setup can dominate overall query capacity on the server platform, so there is some benefit in using a simply adequate crypto algorithm in place of a more involved algorithm. In most public Internet applications, the TLS channel crypto algorithm does not need to have the same cryptographic strength as that used in long term document archival, for example.

It's also the case that for most server platforms the default system settings are not optimal for high capacity DO53 service nor for high capacity DOH service. While for Do53 TCP is largely a low volume activity that may not merit specialised tuning, DoH servers will certainly benefit from carful tuning of both the operating system platform and the TCP stack. In some cases, there looks to be some potential benefit in offloading the TLS into hardware support, and even using a TCP front end grafted on to a private fragmentation-tolerant UDP-based DNS service backend.

Chrome is the dominant browser in the Internet these days, and DoH in Chrome is now on a path to being the default DNS flavour in the browser (https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html). It creates an operational challenge for ISPs, where as soon as they allow a DOH interface to their recursive resolver service to be listed within the Chrome whitelist, in that the load patterns will then immediately jump to match the full load profile from the entire set of Chrome browsers within the ISP's user collection.

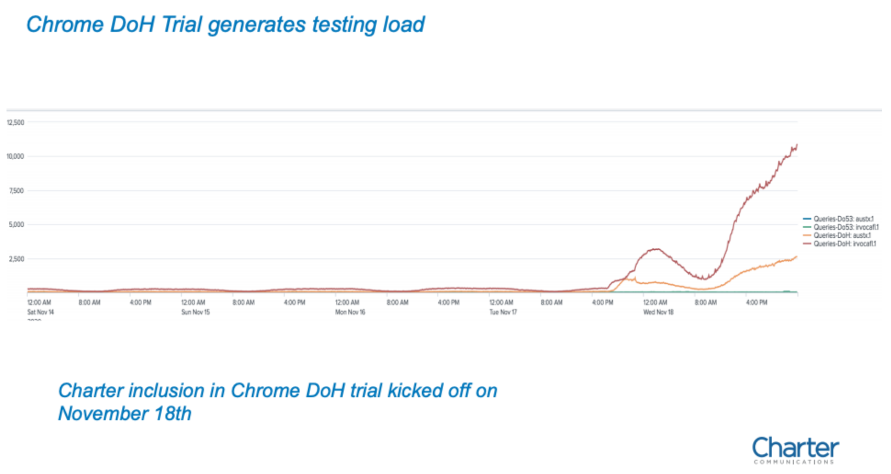


Figure 3 – Chrome DoH load
https://indico.dns-oarc.net/event/37/contributions/807/attachments/793/1403/Spectrum%20DoH_OARC%2034v3.pdf

DoQ Experience

The HTTPS world has been concentrating on QUIC for some years now. There appears to be some strong points in favour of lifting the transport protocol up into the application space and protecting both the transport protocol headers and the content payload within a session level encryption. The observation is that if QUIC is attractive to the environment of web services then QUIC should be equally attractive to the environment of DNS over HTTPS services.

However, in the web world QUIC has been effectively swallowed by HTTP/3. The distinction between QUIC and HTTP/3 is challenging to clearly describe, but the intention is evidently to delineate HTTP/3 as a binding of HTTP semantics to an on-the-wire transport protocol whereas QUIC is best seen as the on-the-wire protocol even if the protocol engine is embedded in the application.

So DNS over QUIC, (DoQ) is strongly similar to DNS over TLS (DoT), whereas DNS over HTTPS is evidently heading to encompass both the layering of DNS over HTTP over TLS over TCP and DNS over HTT over QUIC. From the perspective of DNS over HTTP, the transport used by HTTP us a matter for HTTP, not DNS. Clear? I thought not!

DNS over HTTP/3 carries the DNS query as a HTTP POST command, using an HTTP Header Frame and an HTTP Data Frame to carry the DNS query, and the response is similarly framed. There is a size overhead of between 16 to 32 bytes, plus any HTTP cookies that may be included. There are also the exposure issues associated with HTTP. This leads to a view that DoT and DoQ are preferred starting points when the desire is to use a capable channel encryption technology to support an "overlay" DNS resolution environment. On the other hand, if the desire is to interweave DNS queries within existing HTTP transactions, then the addition of the HTTP framing provides some value. I guess that this leads to a *rule of thumb* that DoT and DoQ are more suited to an operating system platform implementation, while DoH is perhaps better matched to an application implementation.

There has been limited experience with DoQ implementation in the wild so far. One of the early open resolvers to support DoQ is AdGuard, and Andrey Meshkov talked about their early experiences with this service.

It is very early days for DoQ and the relative proportion of use of DNS transport protocols is shown in Figure 4.

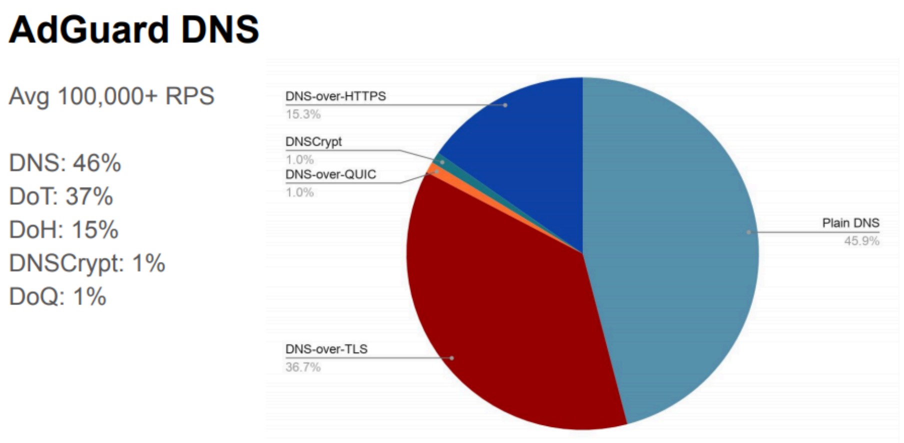


Figure 4 – DNS Query transports as seen by AdGuard
https://indico.dns-oarc.net/event/37/contributions/809/attachments/788/1381/doq_first_experience.pdf

Part of the reason for the scant use of DoQ is the lack of integration into client platforms. Android supports DoT and Chrome now has support for DoH while DoQ and DNSCrypt both require the installation of client-side DNS tools. As we've seen many times, only a few enthusiastic users will play with the default installation settings in a platform, so deviation from the default is relatively uncommon. The users who have redirected their DNS queries to AdGuard have already been willing to play with the default settings on their device. However, there is a far smaller set of users willing to install third party tools on their devices, particularly when those tools are still open-source software projects.

Even so, it's encouraging to see this level of innovation in the DNS and the willingness of open DNS resolvers to support these various efforts.

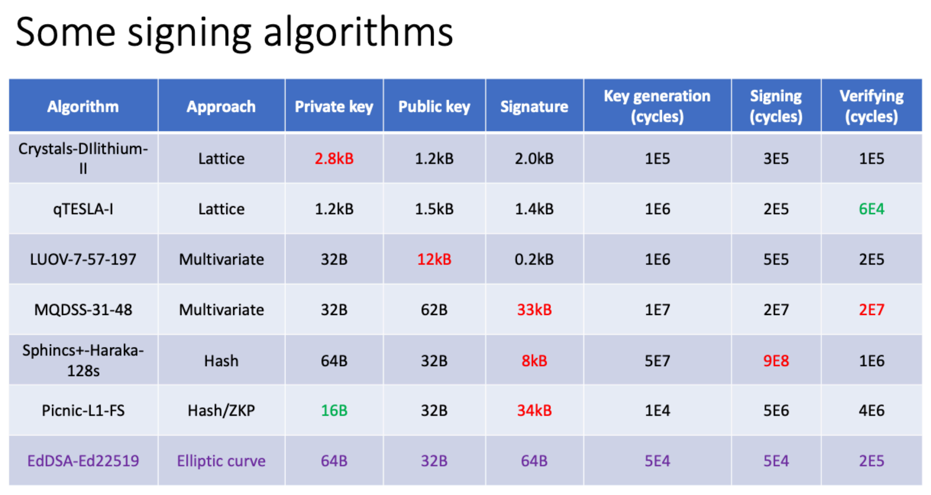
Post-Quantum Cryptography and DNSSEC

We haven't built any large quantum computers yet, but there is already much material out there to be concerned about this topic. The material that is of interest to cryptographers is the change in performance of search algorithms, as most cryptographic algorithms are not into impossible to solve, just infeasible, due to the massive size of an associated search space. Grover's algorithm is a quantum algorithm that can perform a search in $O(\sqrt{t})$ where a conventional algorithm performs in O(t). Shor's algorithm can be used to perform a prime factor search in a time that is almost exponentially faster than the most efficient known classical factoring algorithm ($e^{at} \Rightarrow t^{b}$) The existing prime number and discrete logarithm cryptographic algorithms are vulnerable to quantum algorithms in that computationally infeasible problems become feasible.

The US National Institute of Standards and technology (NIST) has been working on developing a standard for a quantum computing world, looking for candidate key encapsulation methods and signature algorithms that are computationally infeasible in a quantum computing environment. In mid 2020 NIST announced that its plan to release the initial standard for quantum-resistant cryptography in 2022 (https://www.nist.gov/news-events/news/2020/07/nists-post-quantum-cryptography-program-enters-selection-round).

While predictions of the development of quantum computing vary considerably, there appears to be some consensus that quantum computing at scale will be accessible to code breakers in the 2030's.

Of concern to the DNS world, and DNSSEC in particular, is the size of the keys and the signatures, as these are carried in DNSSEC payloads when we shift to using new crypto algorithms that are more resilient to code breaking from quantum computers. The computational load to generate, sign and validate is also of interest. There are a number of approaches that are being explored in the NIST evaluation program, and a summary of the essential properties of these candidate algorithms is summarised in the slide from SIDN Labs' Moritz Müller's presentation on this topic.



Some signing algorithms

Algorithm	Approach	Private key	Public key	Signature	Key generation (cycles)	Signing (cycles)	Verifying (cycles)
Crystals-Dllithium-II	Lattice	2.8kB	1.2kB	2.0kB	1E5	3E5	1E5
qTESLA-I	Lattice	1.2kB	1.5kB	1.4kB	1E6	2E5	6E4
LUOV-7-57-197	Multivariate	32B	12kB	0.2kB	1E6	5E5	2E5
MQDSS-31-48	Multivariate	32B	62B	33kB	1E7	2E7	2E7
Sphincs+-Haraka-128s	Hash	64B	32B	8kB	5E7	9E8	1E6
Picnic-L1-FS	Hash/ZKP	16B	32B	34kB	1E4	5E6	4E6
EdDSA-Ed22519	Elliptic curve	64B	32B	64B	5E4	5E4	2E5

Figure 5 – Post-Quantum crypto algorithm properties
https://indico.dns-oarc.net/event/37/contributions/811/attachments/783/1374/pqc_dnssec_oarc.pdf

Ideally the size of a DNSSEC signature should be under 1Kb, and a conventional processor should be in a position to generate some 100's of signatures per second and validate signatures at a speed that is roughly 10 times the signing speed, or 1000's of signatures per second.

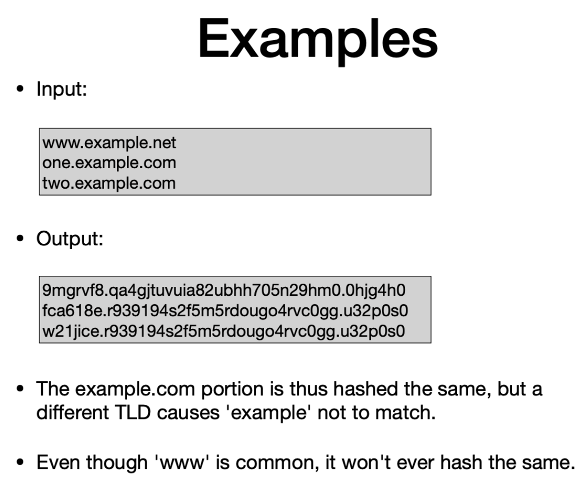
They report on a study of three such algorithms and compare them with the performance of RSA-2048, and ED25519. In two of the evaluated algorithms the public key size is considerably larger than current algorithms, but one evaluated algorithm, Falcon-512, operates within parameters largely similar to one of the ED25519 elliptical curve algorithm in use today.

Hashed RPZ

Much has been done in recent years on improving aspects of privacy in the DNS, but there has not been all that much that has been done in content privacy. Is it possible to use the existing DNS name resolution infrastructure yet keep the DNS labels that you query a secret? And here I'd like to include the recursive resolvers and the authoritative servers into the set of parties who are not privy to my queries. A pointer to a possible approach to the rather extreme level of privacy was provided in a presentation by Jeroen Massar. In Switzerland ISPs are required to block certain Internet content, and the block list is published as a list of domain names contained in a set of files maintained on a SFTP server.

The standard solution to this requirement is to load these domain names into a Response Policy Zone file (https://dnsrpz.info/). However, the additional requirement was to obscure the list of blocked content domain names, and the solution used was to substitute an encrypted hash of the label. For example, the domain name `www.example.com` would be transformed into the domain label `base32hex(hash(www.example.com)).base32hex(hash(example.com)).base32hex(hash(com))`. In the case described by Jereon the hash algorithm used was BLAKE3, keyed with a passphrase. An example of the results of this process is shown in Figure 6.

The deployment of this encrypted RPZ data is performed in a Golang DNS library, where the query name is similarly encrypted with the same hash function, and the RPZ action is triggered if there is a match.



Examples

- Input:

```
www.example.net
one.example.com
two.example.com
```

- Output:

```
9mgrvf8.qa4gjtuvuia82ubhh705n29hm0.0hjg4h0
fca618e.r939194s2f5m5rdougo4rvc0gg.u32p0s0
w21jice.r939194s2f5m5rdougo4rvc0gg.u32p0s0
```

- The example.com portion is thus hashed the same, but a different TLD causes 'example' not to match.

- Even though 'www' is common, it won't ever hash the same.

Figure 6 – Hashed RPZ example
https://indico.dns-oarc.net/event/37/contributions/818/attachments/785/1410/DNSOARC34-HashedRPZ-massar.pdf

I can't help but speculate how it could be used in a more general DNS privacy context. If a zone, and presumably all descendant zones, had its labels similarly encrypted using this mechanism, and then passed to a conventional zone publication service, then the DNS would serve encrypted names in response to encrypted queries. Clients who have knowledge of the passphrase would encrypt the relevant labels of their query and pass it into the DNS in a conventional way. The responses could be resolved and cached in the usual way by the DNS. At this point only the client is aware of the original name that was queried, and no other party would be in a position to see the name being queried. It would be a lot like the IDN implementation in the DNS, where the DNS uses the punycode-encoded labels, while the application has a translation shim that maps between Unicode and this encoded equivalent.

SVC, SVCB and HTTPS

The DNS is used to bind a name to an address, or to use a slightly different terminology, bind a service to a server. What if I want to bind several servers to support to named service? Well in theory I could add multiple address records to the DNS entry and hope the DNS performs the load balancing correctly. What if I want to nominate a primary server and one or more backup servers? How can I shift a service to a new server? How can I place versus services all associated with the same name on different servers? It's clear that as we explore this place and want to associate a richer set of responses from the same base service name the DNS basic binding of name to address just isn't a rich enough construct.

In February 2000 the IETF published RFC2782, the DNS SRV Resource Record. This allows name indirection to associate a named service with one or more named servers, together with a priority to distinguish server preference. This construct was taken up with SIP and LDAP and the TLSA variant is used by DANE, but the HTTP client ecosystem was not an enthusiastic adopter. For many years the hosted service business relied on the CNAME construct where the service name is mapped into a CDN-specific name, and a further mapping is constructed that binds the generic CDN name to a location-specific service host. CNAMEs have some limitations, including an inability to CNAME at the apex of a zone, and the ambiguity when two or more CNAMES are used. SVC records can address these limitations.

But SVC RRs are still not enough. TLS 1.3 introduced Encrypted Client Hello (ECH) and now we would like to bind a service host and a ClientHello public key together. This motivated the specification of a Service Binding Record (SVCB). There are two forms of the SVCB record, the alias form, which is similar to a CNAME record, but allows aliasing at the zone apex. The second form is the service description form, which allows a set of service parameters to be bound to the service profile. A specialised form of the SVCB RR is the HTTPS resource record. This HTTPS RR avoids the underscore prefixes used by the SVCB RRs, allowing clearer mapping with the DNS' use of wildcard domains, and is syntactically well aligned to the CNAME RR. HTTPS records also use two forms, the alias form that aliases a service name to the name of a service point, similar to the CNAME and NAME constructs, and a service form that provides specific service binding information, such as port numbers, priority, and client hello key. a client is expected to issue A, AAAA and HTTPS queries simultaneously. If the response is an alias mode answer, the client must query the alias target name with A, AAAA and HTTPS queries. Then the client needs to resolve the service target name for an IP address (A and AAAA) and use the service parameters to then return a binding profile for the client application to use.

SVCB and HTTPS is still a draft specification (draft-ietf-dnsop-svcb-https), but there is some optimism that publication is imminent.It has been adopted in Apple platforms (iOS 14 and MAC OS 11,and Chrome and Firefox have support for this RR in beta. The DNS platforms from Bind, Unbound and PowerDNS also have support for this RR in development.

According to Ralf Weber from Akamai, the HTTPS RR can reduce the DNS query count considerable, as it can wrap up the aliasing and the IPv4 and IPv6 service addresses into a single response and provide additional service control parameters as well. It takes some of the signalling that was contained in the Alt-Svc HTTP header, which can only be used on the second visit) and place it in the DNS, allowing first time use. For example, advertising the availability of a service over the QUIC protocol was performed as a HTTP header, so that the client had to perform a HTTPS over TCP connection first in order to receive an indication that the same service is available over QUIC. Placing the same information in an HTTPS DNS RR allows the service to be advertised as part of the initial DNS lookup.

It's still early days and it’s not clear if this service record will be widely adopted. It has the potential to significantly improve the connection experience for clients and make improvements in the performance of hosted content for CDNs and allow greater diversity of hosting solutions for content publishers. It allows a richer set of control parameters to be used between content and service providers and content and service publishers, and equally allows the client to be provider with a set of connection parameters prior to the connection.

So, if common sense and a shared motivation to improve the speed and versatility of connection establishment drives adoption then this particular refinement of the DNS deserves to be adopted universally. However, it is a complex record, and our experience in populating the DNS with simple address records have not been all that inspiring. One can only hope that the protracted delay to include the DNSSEC DS and DNSKEY resource records into DNS provisioning and the ongoing issues with DNSSEC key management practices will not enjoy an encore with protracted delays with the adoption of the HTTPS record. It would be good to think that common sense will win out and we will make extensive use of this rather useful function, but personally I find it very hard to be an optimist on this particular topic given a somewhat painful history of adoption of other DNS technologies that require changes in the input zone files of the DNS.

XDP

CPUs are everywhere and today's computers could be more accurately described as a collection of processing engines, rather than a single processor. Network interfaces haven't been built using simple UARTS (*universal asynchronous receiver-transmitters*) for decades and these days contain dedicated processors that control packet handling. A convenient abstraction of specifying packet handling in these data plane processes is the P4 language, a means to specify custom packet header parsing logic.

Linux provides a similar low-level programming facility for the data path, XDP. It's described as a packet processing service at the lowest point in the software stack. XDP code is executed in the NIC driver just after interrupt processing and prior to any memory allocation. An XDP code path is invoked for every packet, and it can be allowed to alter the packet data. The code returns an action selector that can cause the packet to be passed into the host, dropped, bounced back to the NIC or sent onward to a user space socket.

Why is this relevant to the DNS? Part of the issue with the DNS is its use of UDP and the issue with UDP is the ability to mount large scale denial of service attacks that attempt to overwhelm the server using simple techniques of source address spoofing and reflection. Many DNS servers use techniques such as response rate limiting to throttle incoming query rates, attempting to preserve the availability of the service in the face of large-scale attack. The aim is to detect the assumed hostile incoming queries and impose a drop filter to throttle the attack while still allowing the service to operate normally for all other clients. What we would like is an extremely fast mechanism to detect incoming DNS queries and apply some form of rate limiting filter on these packets and do so cheaply and quickly without imposing additional loads on the server.

And this is where XDP comes in. XDP can examine incoming packets, extract DNS queries and device to discard the packet directly or had it to a user space name server. NLnet Labs' William Troop provided a very impressive demonstration of a DNS incoming query rate limiting filter using XDP.

The need for these kinds of packet processing techniques is an illustration of the dilemma facing operators of DNS services. They are essentially required to be promiscuous listeners, which means that they are supposed to accept queries from any and all sources. But this exposes them to various forms of hostile DDOS attack. The only realistic defence we have against such attacks is to absorb the packet flow in a way that does not impair normal service, and the best we can achieve this is to use as little of the server's resources when responding to each query. For this reason, XDP appears to provide a path to increase a server's DNS processing capacity though lightweight packet handling. This approach is attracting interest from DNS server vendors. For example, The Knot resolver includes an experimental XDP module that will be loaded into Intel's multiqueue NIC cards (details at https://knot-resolver.readthedocs.io/en/stable/daemon-bindings-net_xdpsrv.html)

DNS Security Records

There are a collection of DNS Resource Records that are used to protect the DNS payload and to provide some mechanisms for applications to protect the integrity of the application transaction, This collection of RRS includes the following:

- CAA (Certification Authority Authorisation)
  A hint (directive) to CAs that permit only the listed CAs to issue certificate for this domain and any subdomains. Meant to reduce the possibility of collateral damage as a consequence of a breached CA.

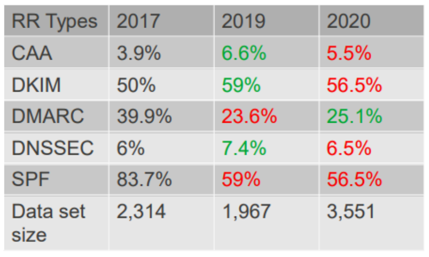
- DKIM: (Domain Keys in Mail)
  An email authentication method designed to detect forged sender addresses in emails. The DKIM RR contains the domain public key used to verify the integrity of the sender's domain and the mail message itself.

- SPF: (Sender Policy Framework)
  Another e-mail authentication method that lists the authorised IP addresses used to send mail with envelope-from addresses in that domain.

- DMARC (Domain-based Message Authentication, Reporting and Conformance)
  A policy setting to indicate if the sending domain is using either DKIM or SPF in outgoing messages.

- DNSKEY
  The public key(s) used to generate DNSSEC RRSIG RRs in the zone.

How are things going?



RR Types	2017	2019	2020
CAA	3.9%	6.6%	5.5%
DKIM	50%	59%	56.5%
DMARC	39.9%	23.6%	25.1%
DNSSEC	6%	7.4%	6.5%
SPF	83.7%	59%	56.5%
Data set size	2,314	1,967	3,551

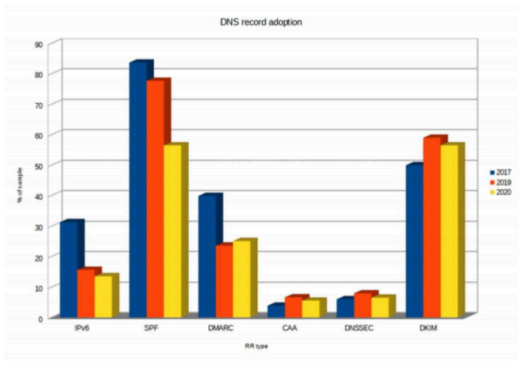


Figure 7 - Update of Security-related DNS RRs over the past 3 years
https://indico.dns-oarc.net/event/37/contributions/817/attachments/789/1379/nominet-oarc-DNS%20survey-rm-20210202.pdf

Their conclusions are not encouraging, in that the update of various security measures appears to be generally quite poor. It appears that there are few resources that promulgate and encourage better practices in these security measures. SPF and DKIM RRs are more widely used, perhaps due to the common use of discard filters for mail that does not use either SPF or DKIM headers. Similar measures have not been adopted for DNSSEC or CAA RRS, so there is little in the way of motivation to drive adoption.

OARC 34

The presentation packs and related material for the OARC 34 meeting can be found at https://indico.dns-oarc.net/event/37/, and if you like you DNS as a TV channel then you can find these presentations at https://www.youtube.com/DNS-OARC.

If you have been working in the DNS then please consider talking about it with your fellow DNS practitioners at an OARC meeting. The next OARC meeting will be held on 6 – 7 May 2021, and the call for presentations is already open (https://indico.dns-oarc.net/event/38/abstracts/).

Disclaimer

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