February 2023
Geoff Huston

OARC 40

OARC held a 2-day meeting in February, with a set of presentations on various DNS topics. Here's some observations that I picked up from the presentations in that meeting.

Cache Poisoning Protection Deployment Experience

In a world where every DNS name is DNSSEC-signed and every DNS client validates all received DNS responses, we wouldn't necessarily have the problem of DNS spoofing. Even if we concede that universal use of DNSSEC is a long time off, we could still achieve a similar outcome if every DNS query was performed across an encrypted channel with server authentication. However, because the overwhelming majority of DNS names are not signed, and because most DNS query/response transactions use open channels, it's still possible to inject false answers into the DNS and thereby deceive unwitting end clients. One could perform this injection close to the edge of the network and impact stub resolvers to mount relatively precisely targeted attacks. A broader attack could be mounted by passing falsified responses to recursive resolvers, who would then pass this data to its clients. In this respect Google's Public DNS service represents an attractive target. The presentation from Puneet Sood and Tienhao Chi from Google is an update on an earlier presentation on the possible measures that Google could use to mitigate the risk of cache poisoning.

There is little a DNS recursive resolver can do against on-path attacks if the name is not DNSSEC signed. However, it's not even necessary for the attacker to be on the path. It's possible for an attacker to force a target resolver to pose a DNS query and then the attacker attempts to forge a first response directed to the resolver that matches the original DNS query and query ID and matches the UDP address and port fields. To counter this risk RFC 5452 suggests that a resolver should use random query ID and varying (and randomly chosen) source port values on its queries. More recently, RFC 7873 and RFC 9018 propose DNS Cookies as another measure to counter IP address spoofing. Authoritative DNS over TLS or DNS over QUIC would also help here, as spurious responses can be readily rejected by the resolver in such a case.

Google has implemented the RFC 5452 countermeasures and DNS Cookies as part of its objective to mitigate such attacks. With respect to the use of DNS Cookies, their measurement of name server capabilities show that cookies are not generally supported as intended. Primarily, the problem appears to be non-compliant authoritative servers returning incorrect responses for DNS Cookies. Google has been performing a structured regular measurement to ascertain the level of DNS protocol compliance performing a daily query of the top 1M nameservers (as per Google's internal ranking). Their findings from this study are that DNS Cookies are only supported in 40% of these name servers, and only supported in 12% of the query traffic.

Consequently, they've applied a number of measures that have been proposed in the past. One is based on a 2008 expired draft, draft-vixie-dnsext-dns0x20-00, which proposed randomisation of the character case in the query label. The off-path attacker has to match the case of the query name used in the unseen query. Of course, this randomisation is most effective in longer query names. The good news is that case integrity, where the case of the query name is reproduced in the query section of the response, is

supported by some 99.8% of all surveyed authoritative nameservers, so this is a useful additional to the approaches to minimise the risk of cache poisoning in off-path attacks. A small number of authoritative nameservers do not preserve the case of the query name in the response, and some 2,000 nameservers are on a disabled list. Some servers exhibit case randomisation only sporadically. Google's public DNS resolver will retry the query over TCP if there is a case mismatch between the query and response.

DNS Cookies allow a client and server to exchange pseudo-random values across successive query/response interactions. They essentially provide a weak form of authentication of DNS responses, useful in the context of an off-path attacker. Google's experience with DNS Cookies shows limited support of cookies among name servers. There are some issues with anycast IPs and load balancing across DNS server farms. Without cookies there is no requirement to pass successive queries to the same server, while cookies impose a more constrained behaviour on the front-end load balancers with case randomisation, the mitigation of server cookie failure is to retry the query over TCP.

Google also performs unilateral probing for support of Authoritative DNS over TLS (ADoT) support. Such TLS queries are far more challenging for an attacker, and the hope here is that the cost of any form of successful attack on DoT is prohibitively high. ADoT is in use by some 700 name servers and covers some 4.5% of Google's egress traffic. ADoT has a comparable average latency and comparable response success rate to DNS over UDP. There are some residual TLS issues, such as early session closure from the server, even to the point of query interruption.

Neither DNS cookies nor query name case randomisation can protect a resolver against on-path attacks. ADoT provides a more robust defence mechanism, but it comes at a considerable cost to the DNS infrastructure. We quickly get back to an age-old attack mitigation question: What is the frequency and impact of such off-path substitution attacks, and is the incremental cost of such mitigation measures commensurate to the cost of successful attacks? Expensive defences for low-volume low-impact esoteric attacks don't make an awful lot of sense, while not making small changes to mitigate high volumes of attacks is also neglectful.

Maybe it's worth reviving the 15 year-old specification for case randomizing of the query name. It seems like a cheap measure that would make the task of the off-path attacker more challenging.

Post-Quantum Crypto

Quantum computing at scale is still some time away, but the crypto folk have been looking at crypto algorithms that can resist efforts to crack the algorithm using quantum computers. The US National Institution of Standards and Technology (NIST) have been working on this issue for some years now, and in July 2022 NIST announced four candidate algorithms, CRYSTALS-KYBER, CRYSTALS-Dilithium, Falcon and Sphincs. Falcon will be standardized for use in cases where the CRYSTALS-Dilithium signatures are too large and Sphincs will be standardized to avoid sole reliance on the security of lattices for signatures. The presentation on this topic from Verisign's Andres Fregly looked at this effort from the perspective of the application of these algorithms in DNSSEC.

The implications of the extended signature size (up to 7,856 bytes in the case of SPHINCS-128s) are certainly extremely challenging for the application of DNSSEC in DNS over UDP transports. This has prompted some interesting proposals, including the packaging of a large signature in a Merkle Tree Ladder, which can then be delivered in parts where individual components will fit within a 512 byte payload constraint.

I can't help wondering if this mix of PQC, DNSSEC and UDP is just a mutually incompatible scenario. In the domain of web content these payloads are tiny, and we know that the network infrastructure can pass these signatures across streaming transports with ease. So why are we still trying to jam these signatures from a 2020's crypto algorithm into a 1980's datagram UDP transport?

Perhaps it time to turn off the DNSSEC OK bit in DNS queries over UDP transport, and look at how to optimise the performance of DNSSEC, including validation, in a context that is deliberately constrained to reliable streaming transports.

Operational Experience with DNSSEC-signed zones

From the very early days of the Internet, it was observed that there was only one issue with the Internet, namely "scaling", and all the other issues were derived from that. Now that may be a gross simplification, but there is an element of truth there, and DNSSEC is no exception. Shumon Huque presented on operational experience with DNSSEC-signed zones, and the experience related to size-based issues.

When a zone file contains tens of millions of records, then the resource record count of the zone increases significantly with an NSEC/NSEC3 record per name, plus its RRSIG signature, and an RRSIG per RRSET for each name. For a zone that is dominated by CNAME records then the RR count will increase to 4 times the record count and 15 – 20 times the size of the zone file. If you are looking for a DNS provider to serve your zone, then when the zone gets to such a size there are few providers who can cope with such a load.

Large zones also tend to experience a high incremental change level, requiring incremental signing and incremental zone transfers, which increases the zone update traffic load. This requires consideration of the signature validity window, and the desire to stagger the expiration of individual signature records to avoid large update spikes. How to cope with further growth remains a relevant question, and the longer-term approach is to collapse a large collection of records into a far smaller wildcard record set that points into a CDN-hosted database for the wildcard mappings.

To avoid the pitfalls of IP fragmentation they use Elliptic Curve signature algorithms and cap the UDP message size to below 1,500 bytes and force the use of TCP for larger responses.

They have encountered some DNSSEC implementation bugs, generally associated with negative responses. The combination that appears to have triggered these behavioural anomalies is a negative response generated by an online signer. The bitmap of defined RRtypes may not be accurate, and clients who cached this reduced type set subsequently provided negative answers to defined records.

The lessons from this experience are sobering. Even if you outsource your DNS operations to others you still need to retain substantial DNS expertise yourself. You need to play very close attention to the way denial of existence is handled. Shumon also recommended to use multiple DNS providers where you can. I'm not completely convinced about this last recommendation. Ever since we encountered the issue with DNSSEC validators being overly persistent in seeking a positive validation outcome, many validators will cease as soon as they encounter an invalid state. They do not exhaustively interrogate all of the providers to see if there is a different outcome from an alternate server. Coordination key distribution and key rolls across multiple providers can also introduce operational fragility.

Is the DNS "IPv6 Ready"?

Akami's Ralf Weber made an interesting presentation on a data-gathering exercise looking for the extent to which IPv6 was configured into the name server environment of the DNS.

The dataset used in this work was based on an anonymized set of Akamai customer DNS logs, and this name set was passed through a resolver to find the zone cuts and then find the name servers for the "terminal" name. These name servers were tested for reachability over IPv4 and IPv6. The exercise started with a 1-week query long that was reduced to 2B unique fully qualified domain names. Interestingly, the customer query log contained 57M unique delegated domains, of which 45M are the result of singular hits. Out of the 300B queries less than 2B queries (or less than 1%) yielded 45M singular domains (or 78%). It seems that we all ask the DNS the same questions most of the time!

The result of this work is that 53% of domains are resolvable over both IPv4 and IPv6, and 47% are IPv4 only. Some 300 domains out of the 80M domain set are IPv6-only.

Ralf's conclusion from this data is that the DNS is not IPv6 ready. I think that this depends on what you define as "IPv6 ready". The 53% dual stack rate in DNS domain servers indicates that IPv6 deployment is proceeding at a rate that is slightly greater than the level of IPv6 deployment in consumer access networks. If you take an entire region that has minimal IPv6 use so far, and Africa is a good example of this situation, then there is little in the way of motivation or local technical experience to deploy IPv6 across local DNS services outsize of the handful of countries that have more than cursory levels of IPv6 use. So, what we are seeing is a whole set of local micro-decisions as to when and how to shift from IPv4-only to dual stack. The timing of these local decisions to undertake this first part of the larger IPv6 transition is up to each party, as IPv4-only and dual stack clients can still access the entire set of IPv4-only and dual stack servers. Any client or server who drops IPv4 support today cannot reach or be reached by some 70% of the Internet's user base. What do we mean by "IPv6 ready"? Is it the point when a client or a server can complete their transition and drop all IPv4 support to become IPv6-only? If this is what we mean by "IPv6 ready" then no, nothing in the Internet today is "IPv6 ready", including the DNS.

If we decompose this question and look at the extent to which we've transitioned from IPv4-only to a dual stack scenario that supports network transactions over IPv6, then it seems that the transition of servers from IPv4-only to dual stack has been undertaken for slightly more than one half of the domain name servers. Perhaps slightly over one half of the DNS server infrastructure is "IPv6 ready".

The DNSSEC Path Not Taken

One of the interesting aspects of DNSSEC's design was to tightly bind the hierarchical structure of the name space to the trust space. Why should I trust that the key used to sign records in a zone file is the genuine article? Because the zone's parent has signed across the public part of this key with its private key. Why do I trust that the zone parent's key is genuine? Because its parent has signed across this key, and so on until we reach the Key Signing key of the root zone. Why do I trust this key? Because it was configured into my DNS resolver as a trust anchor. This means that the entire DNSSEC framework operates with a single trust anchor.

To achieve this binding, we placed the public keys directly into the DNS (as DNSKEY resource records). The DNS zone operator who controls the content of a DNS zone also controls the DNSKEY record and the zone's signing keys. There is no need for a third party to attest that this zone operator is the "genuine" operator. That attestation is provided by the zone parent who has delegated the zone and confirmed by its signing across zone's key.

The alternative here is to use the X.509 public key certificate structure. Here a party applies to a Certification Authority (CA) and if it can fit within the CA's certificate issuance policies, and satisfy a number of prescribed tests, then the CA will use its private key to sign across the applicant's public key, and format this signature into the form of an X.509 certificate. Any client, or relying party, who trusts that the CA never lies and never deviates from its policies and tests, can then trust that the party who controls this key has met the CA's issuance policies. So, if the CA policy is that certificates are issued to domain zone administrators who can show that they control a domain zone, then this X.509 certificate can play a similar role to the key-signing chain used in DNSSEC.

And this is exactly the thought experiment described by a research group at the Seoul National University in a presentation "Guaranteeing the integrity of DNS records using PKIX certificates". The problem with this X.509 approach is the same as the problem with the existing collection of Domain Name certificates. The issue with multiple points of trust in a PKI is that the client, or "relying party" cannot know in advance which CA has certified the keys used to sign a given domain zone. If any CA issues a

certificate under adverse conditions (hacked server, social engineering, malfeasance, corruption) then the relying party may be deceived into believing the wrong keys and trust incorrect DNS information. The overall trust model of this kind of distributed trust framework is akin to the "weakest link" where the strength and integrity of the system is only as strong as the weakest or most vulnerable CA.

Even if an incorrectly issued certificate is detected, how is it marked with "do not use under any circumstances" label? The problems with timely certificate revocation have been a sore point for domain name certificates for many years, and this proposal presents no new ideas as to how to manage this issue.

Perhaps it's worthwhile to revisit these basic design decisions of security frameworks to ensure that we haven't missed anything that we should've been aware of. But it was up to me I would look at the way DNSSEC validation requires the assembling of data by the client and look at ways that push this burden of assembling the key validation data from the client to the server.

Measuring the Effectiveness of Aggressive NSEC3 caching

Back in 2018 at OARC 28, Petr Špaček reported on the outcomes of his tests in measuring the effectiveness of aggressive NSEC caching (RFC 8198) in deflecting random subdomain attacks back to recursive resolvers. His conclusion was that this measure was effective in deflecting such attack traffic for small and medium-sized zones and was related to the TTL value for very large zones. In a final note Petr noted that NSEC3 was yet to be similarly investigated. Otto Moerbeek from PowerDNS has now filled in this gap, and his results are somewhat surprising.

The *unbound* and *bind* resolver implementations support aggressive NSEC caching for NSEC records only, while the *Knot Resolver* and the *PowerDNS Resolver* extend this support to NSEC3 records, as long as there is no NSEC3 opt-out configured in the zone.

The observation that triggered this work was that the *.nl* zone, a large zone of 6M names of which more than 60% are DNSSEC-signed. When the zone was switched from NSEC3 with opt-out to NSEC3 without opt-out there was an expectation that the NXDOMAIN query rate at the zone's authoritative servers would fall as a result of this switch, but no such drop was observed. Further tests showed that a small zone with NSEC3 signing did deflect NXDOMAIN queries to caching resolvers, while larger zones showed no such effect. Why is NSEC3 less effective than NSEC in caching negative outcomes for larger zones?

There is a critical difference in the span of names that are defined by NSEC and NSEC3 records. NSEC records define a related span of names, such if the names are drawn from a common language set then there is a distinct probability that a small subset of NSEC records would cover a huge span of unused names. In other words, if the names are clustered together then it takes only a small set of NSEC records to define the span of possible query names that lie outside this cluster. Hash functions have the property that small variations in the original data causes large variations in the hash value. So, a clustered set of names would create a dispersed collection of hash values, and there is a far lower probability that a small set of NSEC3 records would span a large collection of non-existent names. This negatively impairs the caching efficiency of NSEC3 records.

This, in turn, highlights a substantive issue in attempting to measure the effective of caching of these types of negative spanning records. When looking at NSEC caching, the distribution of names within the zone and the degree to which the names are clustered together impacts the size of the size of the span of each NSEC record. When measuring the effectiveness of NSEC caching, the distribution of queries is also important. If the query name set is randomly distributed across the name space, then the likelihood that large spans will be used to match many random name queries is high, and the NSEC cache will be highly effective from the outset of the test. If the query name set is also clustered in the same cluster space as the zone's names, then NSEC caching efficiency will likely drop (as the span of each NSEC record within the space where the zone names are clustered will also be smaller).

With NSEC3 records, any name clustering factor is neutralised by the hashing function, and NSEC3 hashing will degrade in efficiency as the number of labels in the zone increases. Which matches the observations seen by Otto in his study.

It has been observed many times that the only benefit for NSEC3 is opt-out, and the case where opt-out may be of benefit is large zones with a low delegated domain signing rate. The issue of zone enumeration prevention with NSEC3 records has been demonstrated to be a weak claim, as the hash algorithm used to generate the resequenced name set can be cracked.

Public Annotations of DNS RFCs

There are many RFCs about the DNS.

The canonical specification of the DNS that is normally cited are the pair of RFCs, RFC 1034, "Domain names - concepts and facilities", and RFC 1035, "Domain names - implementation and specification", both published in November 1987. However, these two core specifications are just the tip of a rather large iceberg. One compendium of all the RFCs that touch upon the DNS lists some 292 RFCs. That implies that to claim that the DNS is essentially unchanged over this forty-year period might be a bit of a stretch, but nevertheless the fundamentals of the DNS have been constant. Those additional 290 RFCs illustrate the observation that we've spent a huge amount of time and effort over these forty years focused on tinkering at the edges!

Even so there are still many aspects of the DNS behaviour that are just assumed, or undocumented in any RFC (such as draft-vixie-dnsext-dns0x20-00, for example). What should we do about this overwhelming volume of specification?

Time for even more documentation, and this time we need tools to create even more documents about the other documents that describe the DNS. This is purpose of the annotated RFC collection. The feature of this compendium is to add annotations to these RFCs. I'm not sure who will do this annotation work, or who will find it useful, and I guess it will depend on the level of curation of such annotations. Well managed, it could be a useful source of clarification of some of the more obscure points of detail in the RFC collection. Left as an uncurated open access space it will suffer from the same level of degradation and abuse as any other write-access public space on the net, sadly.

Dynamic DNS Servers

The conventional paradigm of the DNS operation was that of a distributed database, where a query contains a name that acts as a lookup key, and the response is the information that is stored in the database.

There is the possibility of a different perspective on the DNS, where the query names represent an instruction to the server, and the response is formed by the server executing this implicit instruction. AFNIC's Stephane Bortzmeyer has implemented an experimental dynamic authoritative server that works along these lines. As Stephane points out, this work is not exactly novel, and implementations of dynamic DNS servers have appeared from time to time in the past. In Stephane's case the server implements NSID, DNS Cookies, TCP with pipelining and out-of-order responses and DNSSEC signatures.

In thinking about the possible role of such a dynamic server in the DNS ecosystem, it's clear that this can be a useful tool in certain circumstances. We've used such an approach in APNIC Labs to create a single DNS server that can produce a particular form of DNS response based on the query name. For example, a response may be DNSSEC signed or not, and the DNSSEC signature may be valid or invalid, all depending on the value of subfields within the query name.

Reducing Default DS TTLs

There have been a number of recent outages relating to DNSSEC-signing of a zone where the problem with the zone persisted for many hours because the recursive resolvers had cached the erogenous DNSSEC information, and it was necessary to wait for this cached information to time out. Slack experienced this in 2021, but there are by no means the only party that has been tripped up by DNSSEC. DNSSEC-related outages are a consistent issue these days.

Google's Puneet Sood is advocating that we reduce the default TTL for DS records. This aspect of TTL tuning represents a compromise between offloading query traffic from authoritative servers onto caching resolvers, and the ability to undertake timely changes. Longer TTLs can trap the zone in a broken state that is essentially inescapable until cached entries expire, while shorter TTLs cause a higher query load on the server but allows the zone administrator to make changes in far shorter timeframes.

A broken DNSSEC state renders the zone unresolvable for the 30% of the Internet's users that rely upon DNSSEC-validating resolvers. Allowing repairs to a occur to a broken DNSSEC in a timely manner, even if it is simply a case of temporarily backing the zone out of DNSSEC, seems like a preferable position.

OARC 40

The full workshop program, and all the presentations from the workshop can be found at the OARC 40 web page.

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