The ISP Column
A monthly column on things Internet

August 2014

Geoff Huston

The Cost of DNSSEC

If you're playing in the DNS game, and you haven't done so already, then you really should be considering turning on security in your part of the DNS by enabling DNSSEC. There are various forms of insidious attack that start with perverting the DNS, and end with the misdirection of an unsuspecting user. DNSSEC certainly allows a DNS resolver to tell the difference between valid intention and misdirection. But there's no such thing as a free lunch, and the decision to turn on DNSSEC is not without some additional cost in terms of traffic load and resolution time. In this article, I'll take our observations from running a large scale DNSSEC adoption measurement experiment and apply them to the question: What's the incremental cost when turning on DNSSEC?

There are of course two perspectives in the DNS: that of the resolver, or client, and that of an authoritative name server, so lets look at each of these in turn.

DNSSEC in the Resolver

The first is the perspective of a client of the DNS. What's the cost of switching on DNSSEC validation of DNS responses?

Intuitively, we would expect to see a resolver take slightly more time to perform name resolution, and perform additional queries as part of the response validation process when resolving a name associated with a DNSEC-signed zone. The responses that the resolver will receive will also be larger in this case, as the received responses will include the DNSSEC signature data.

Can we quantify this difference?

Lets look at a resolver's behavior, comparing the situations of with and without DNSSEC.

Firstly, lets look closely at a DNS resolution trace, using a query for a 5-label name when the resolver has no cached state, and is not performing DNSSEC validation (see Log 1). This entire name resolution process took 4 DNS queries, and a total of 0.535 seconds elapsed time. In terms of DNS traffic the resolution of this query generated 292 bytes of outbound query traffic and a total of 1,622 bytes in received DNS responses. However, the queries to the root name server and the *.net* name server was, to some extent, priming the local DNS cache. If we were to make a second query using the same resolver, within a few seconds of the original query, and changing only the terminal label, then the DNS name resolution process is much faster and much more efficient, and the process entailed a single query and response, with a total of 191ms elapsed time, 83 bytes of query and 134 byes of response (see Log 2).

We can compare this to a DNS query to resolve a similar name, but this time the local resolver is configured to perform DNSSEC validation, and the name being queried is DNSSEC-signed (see Log

3). Including the root key priming queries, the resolver's query from a state where the local resolver's name cache is completely empty took 20 DNS queries, taking a total of 2.102 seconds, while sending a total of 1,446 bytes and receiving 13,026 bytes in responses. That's 4 times longer in elapsed time, and 8 times the traffic load compared to the non-DNSSEC case. Once the cache is primed the DNSSEC overhead is far lower of course, and a change in the terminal label requires only a single query (see Log 4). Even so, the DNS response is some 8 times larger in the signed case. Resolver caching is a very powerful tool in the DNS and it certainly improves the performance of the DNS, and far more so in the case of DNSSEC. Table 1 shows a comparison of these cases.

	Queries	Time	Bytes Out	Bytes In
Unsigned	4	0.535	292	1,622
Unsigned, cached	1	0.191	83	134
Signed	20	2.102	1,446	13,026
Signed, cached	1	0.191	82	1,123

Table 1. Resolution of a 5-label DNS name, with and without DNSSEC validation

Serial vs Parallel in Validation Queries

The performance of the Bind 9.9 resolver that was used to generate these logs could be improved considerably by changing the serial nature of the DNSSEC signature validation procedure into a parallel procedure within the DNS resolver. In this case there are 14 subsequent queries that are required to perform the DNSSEC signature validation process, and this takes 12 separate query / response intervals. With just two exceptions, all the queries are queued up in a serial queue and await the successful completion of the predecessor query before being passed out to the network.

However, as is evidenced in separate experiments with various permutations of deliberately broken signature validation configurations, the resolver's key-matching validation process is not initiated until the entire DNS query process for the DNSKEY and DS resource records has completed. A faster approach would be to set off a larger set of these queries in parallel. If Bind were able to parallelize this process, and have, say, 20 outstanding queries at any point in time, the cache priming exercise for a validating resolver would be reduced from 2.102 seconds to 0.725 seconds, or just one query/response interval (190ms) longer than the unsigned non-validating case.

It is also unclear why the Bind 9.9 resolver is directing queries for a zone's DS record to the grandparent before re-querying the parent zone. These queries to the grandparent zone server fail, as the grandparent zone's authoritative service in this case has no knowledge of the DS records of its grandchild zone. If this is some attempt at DNS query optimization it is hard to see precisely what is being optimized here by this behavior.

Can we compare these bench tests against observed experience? As part of a program of measuring DNSSEC deployment across the Internet we've been requesting end clients to fetch two unique (uncached) URLs. One is a DNSSEC-signed name, and the other is not DNSSEC signed.

How much longer does it take for clients to use DNSSEC-validating resolvers to resolve an uncached DNSSEC-signed name, as compared to a non-DNSSEC resolver resolving an unsigned name?

We use a server-side measurements, measuring the elapsed time from the reception of the first DNS query for a particular name to the reception of the HTTP GET command. When we measure clients who use non-DNSSEC-aware resolvers, and compare the relative fetch time for resolving an unsigned uncached DNS name to resolving a signed uncached DNS name we see the cumulative distribution as shown in Figure 1 for "No-DNSSEC". For some 20% of end users the DNSSEC-signed name is resolved faster than the unsigned name. This appears to be part of the experimental variation experienced within the Flash scripting system used in the experimental process, where the DNS resolution process appears not to be tightly coupled with the URL fetch subsystem. Similarly, some 20% of users take longer to process the DNSSEC-signed domain name, even through the resolvers that they are using do not perform any DNSSEC-related RR queries. So the "No-DNSSEC" data series represents the "control" profile in this experiment. The DNSSEC-Validating data series represents the same measurement, performed for users who use DNS resolvers that perform DNSSEC validation. In this case the median point of this second data series represents an additional time component of 300ms to resolve a DNSSEC-signed name, which is consistent with the observation that in this case the additional DNS queries that are performed in this case are a DNSKEY RR query for the terminal zone, and a DS RR query for the KSK and ZSK keys, as reported by the parent zone, and it appears that most resolvers perform these additional queries sequentially. While validation would necessarily involve assembling the chain of DNSSEC key data through to the root key, much of this data is cached by the resolvers used by these clients, as the query rate for the zone keys for the common parent zones are far lower than the query rate for the unique terminal zones.

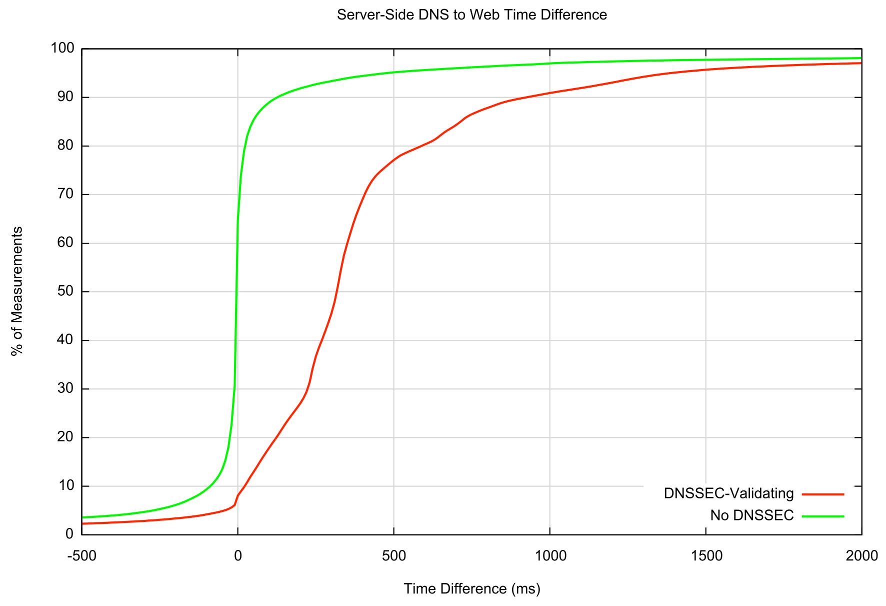
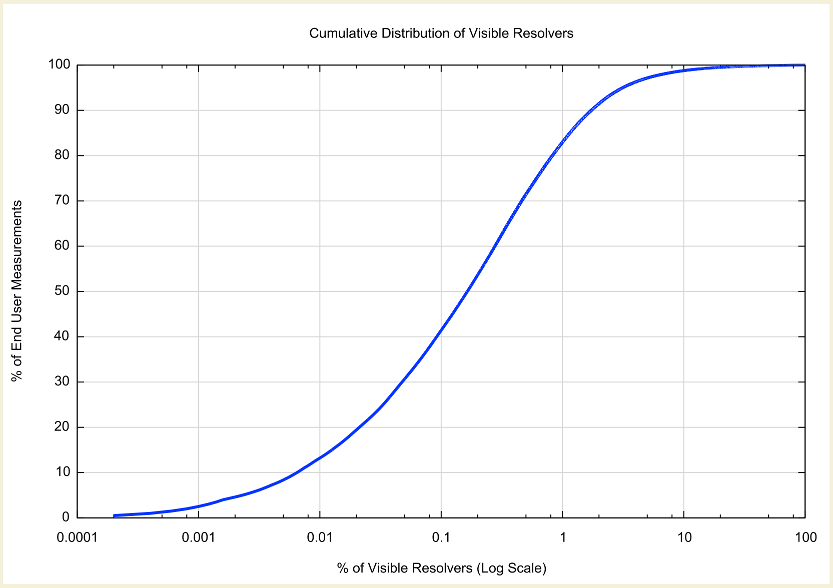


Figure 1 - DNS Resolution Time Comparison

DNS Resolver Concentration in the Internet

DNS caching is extremely effective in the Internet, and arguably the DNS enjoys a singularly high ratio of caching efficiency as compared to any other application or service on the Internet. Part of the reason for this lies in the concentration in the use of a small set of DNS resolvers by very large numbers of end clients.

Across some 40 million individual DNS measurements over the past six months in our experiment, just 100 resolvers are used by 20% of these end user devices, and just 1% of the 500,000 visible resolvers are used by 82% of the end user devices. The implication is that once one of these heavily used visible resolvers caches a response, then a significant proportion of end client devices will take advantage of the resolver's cache.



DNS Resolver Distribution across Clients

It appears that the major "cost" component of DNSSEC for the client is the additional time taken to fetch the DNSSEC credentials, and this cost is exacerbated by the serialization of these signature fetches in most resolvers. However, this cost is effectively mitigated by caching, and the incremental cost to resolve popularly used DNS names for the overwhelming number of end users is minimal due to the two factors of concentration of recursive resolvers and caching. If the end client is relying on the local resolver to perform DNSSEC validation, the result of caching would eliminate such incremental costs for the client. If the client is performing its own validation of DNS responses then the validation component would add an additional query cycle to the local resolver to retrieve the DNSSEC signature data when it is not already present in the resolver's local cache.

We observe that is those cases where the client is able to resolve the unsigned DNS name in a single query (some 80 – 90% of all clients), the additional time taken to perform DNSSEC validation is no more than an additional 500ms for an uncached DNS name. It should be remembered that the data presented above relates to an uncached DNS name. DNS caching effectively moderates this time penalty, so the actual experience would be no worse than the data given here, and likely would be far better.

There is one further potential aspect to the client's experience we should evaluate here. DNSSEC signatures imply large DNS responses, and there is a lingering suspicion that some parts of the Internet still reside behind middleware that expects that the DNS is a UDP protocol where the question and the answer and both no larger than 512 bytes in size. If that's the case then there would be a visible subset of end users who would be unable to resolve the DNSSEC-signed name. Is this visible in the experiment data?

One way to search for such patterns of use is to pick apart the DNSSEC experiment, and find a pattern of behavior that would correlate with an inability for a client to resolve a DNSSEC-signed name. In this experiment we use three URLs. One is unsigned, one is signed with a valid DNSSEC signature and the other is also signed, but the signature validation chain is deliberately broken. If the end user completes the experiment, then if they cannot resolve any DNSSEC-signed name we would see DNS resolution attempts for all three unique URLs, but we would see web fetches for only the URL with the unsigned DNS name. If these end users cannot resolve a DNSSEC-signed name, then we would also expect to see that all DNS queries for the signed name include the EDNS0 extension with the DNSSEC_OK flag set. In other words all queries for these signed names need to elicit the larger signed response.

The problem in the context of this particular experiment we've used to collect DNS data is that we cannot be assured that all users will complete all three URL fetches. The experiment will terminate prematurely when the user leaves the web page that delivered the experiment, and the experiment's control script, Adobe Flash, appears to execute the three fetches in a relatively random order. So we can anticipate a small rate of "drop" for any of the three URL fetches. If there is a signal relating to inability to resolve DNSSEC-signed names, then we would expect that the drop rate associated with the conditions described above to be higher than the drop rate associated with other combinations of drops and fetches. We need a couple of control experiments that will display this background "random" drop patter. The first control is the number of end users who fetch the DNSSEC-signed names, but not the unsigned name. The second is the number of users that fetch the unsigned URL and the badly signed URL, but no the validly-signed URL. The third control is the number of users that fetch the unsigned URL but no the DNSSEC-signed URL, but do not necessarily exclusively have the DNSSEC-OK bit set in their queries. Figure 2 shows the outcome of this measurement over 56 million sample points across 9 months. The three control drop rates are relatively consistent with each other, at around 0.1% to 0.3% of users, while the DNSSEC-resolution drop rate is 0.02% lower than the three controls, and not higher. This data leads to the observation that at this level of resolution there is no discernible signal that would indicate that there exists a population of users who are unable to resolve a DNSSEC-signed DNS name. This data does not reveal any discernable DNSSEC "drop rate" in today's Internet.

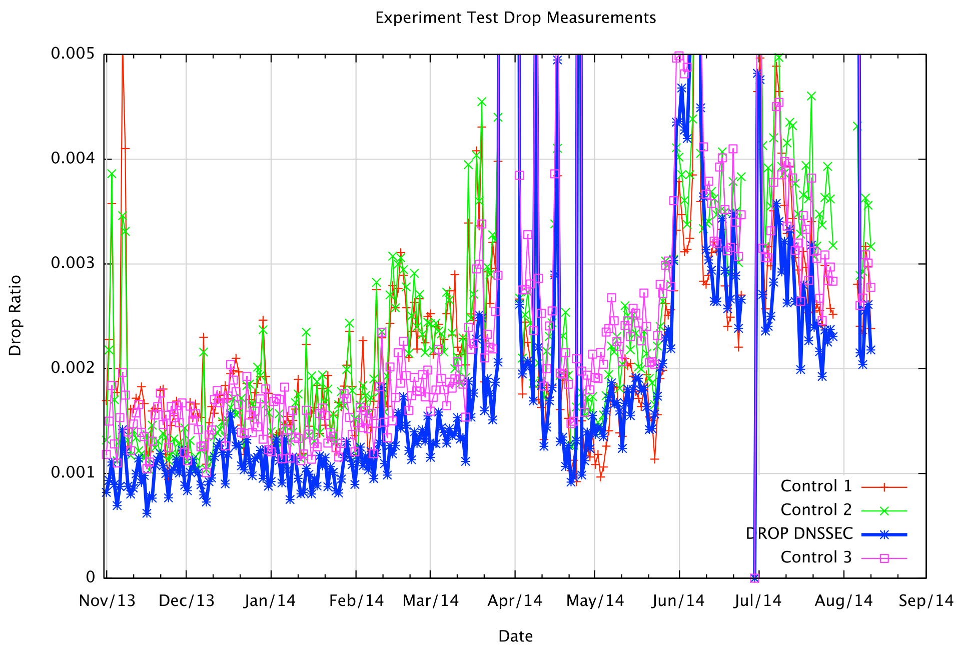


Figure 2 – Drop Rate of DNSSEC-Signed DNS names

There is a caveat here about the limits of this particular experiment. None of the DNSSEC-signed query responses are larger than 1280 octets, let alone 1500 octets. That implies that we are not causing any form of UDP packet fragmentation in either IPv4 or IPv6. So the specific failure point were would be exposing in this particular experiment is clients' resolvers behind systems that enforce a limit of 512 bytes in the maximum size of DNS responses, but where the resolver has enabled EDNS0 in their queries. It would be useful to repeat this experiment using a DNSSEC response size the is greater than 1500 octets, to see whether UDP fragmentation in either IPv4 or IPv6 causes an elevated drop rate due to DNS resolution failure with such larger responses. This may happen in a future run of this experiment.

What's the cost to clients of a resolver when the resolver turns on DNSSEC validation? Taking caching into account, the incremental cost appears to be largely invisible. Cache refresh will take additional cycles, and the refresh time will vary depending on the depth of the label, the state of the local caches and the network path between the resolver and the authoritative name servers.

DNSSEC in the Server

What about the incremental cost for DNSSEC when considering authoritative name servers? What's the difference in load when serving a signed zone, as compared to an unsigned zone? And what predictions can be made when considering the hypothetical situation where every DNS resolver performed DNSSEC validation?

In this experiment we are using a name with 5 significant labels. The 5th label maps into a wildcard in the terminal zone, and the 4th label is intentionally unique, and we are looking at the load at the authoritative name server for these uniquely-named signed zones. The name delegation and key information from the initial three labels is expected to be cached, so the query load being profiled here is effectively that of this 4th zone. For an unsigned zone we see a single query at the authoritative name server, a 128 byte query for the A record, and a 179 byte response.

When the zone is DNSSEC-signed and the resolver is DNSSEC aware there are commonly 2 additional queries, one for the zone signing key in the zone (the DNSKEY Resource Record (RR)), and the second for the Key Signing Keys and Zone Signing Keys, where the query is directed to the parent zone name server (the DS RR). In this case, as the authoritative server is authoritative for both the signed zone and its signed parent, both queries are visible at the authoritative name server. The result is that the authoritative name server now has 3 queries, totaling 287 bytes in queries and 2,251 bytes in responses in the case of our experiment. This would lead us to the conclusion that the cost of serving a DNSSEC-signed zone is an increase in the query load by a factor of 2 (or 3 if the authoritative name server also serves the parent zone), and an increase in the response traffic load by a factor of 11 (or 12 if it also serves the parent zone) when responding to a DNSSEC-aware resolver.

What do we see from this experiment?

In this experiment the authoritative name server was authoritative for both the terminal signed zones and the common parent zone, so we would expect the server to see both the DNSKEY RR query for the terminal signed zone and the DS RR query for the same zone that was directed to the parent zone. The results over a period of 8 months from December 2013 to July 2014 are shown in Figure 3.

The query response traffic for the signed zone have riven from approximately 7 times the traffic for the unsigned zone in December 2013 to approximately 8 times the traffic in July 2014. This rise is most likely due to the rise in the number of DNSSEC resolvers that perform DNSSEC validation over this same period rather than any change in the behavior of individual resolvers. What this data appears to be saying is that if you take an unsigned DNS zone and sign it, then the authoritative name server will see an increase in the outbound DNS response traffic by a factor of around 8 times. Today some 20% of the Internet's client base use DNS resolvers that perform DNSSEC validation. If all resolvers were to perform DNSSEC validation would we see a increase in DNS responses to 40 times the load of an unsigned name? And if that's the case how can we reconcile this data point with a theoretical prediction of around 12 times the traffic?

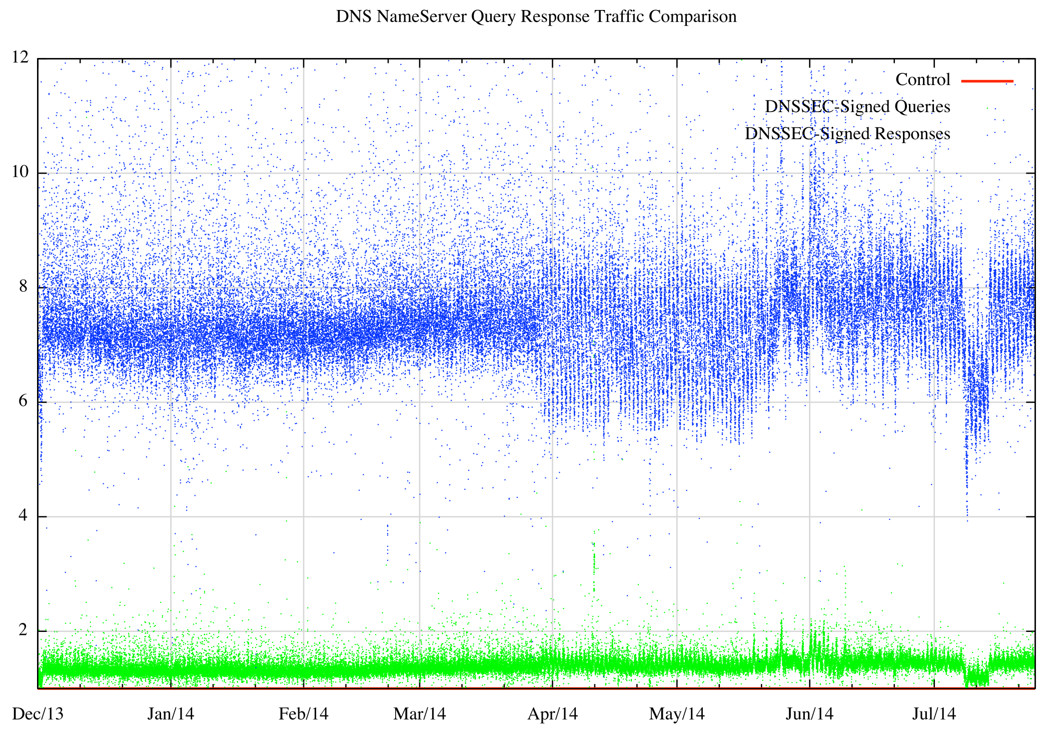


Figure 3 – Query and Response Ratios: Signed vs Unsigned

The answer lies in the use of the EDNS0 extension mechanism, and the use of the DNSSEC-OK flag within that extension. While some 80% of the Internet's clients are using DNS resolvers that do not perform any DNSSEC signature validation, some 85% of clients are using DNS resolvers that include the EDNS0 extension in their query, and they set the DNSSEC-OK bit of this extension. In other words, some two thirds of the Internet's user base are using DNS resolvers that are already requesting DNSSEC signature data to be included in response to their queries, but do not then bother to validate this signature data! So today we have only some 20% of clients using DNS resolvers that perform some form of DNSSEC signature validation, yet due to the widespread use of EDNS0 we already have a 7 - 8 times factor in increased size of query response traffic. If this 20% figure were to grow to 100%, where every resolver performed DNSSEC validation, then the increase in traffic load at the authoritative nameserver would not rise by a factor of 40, but a more modest factor of around 11 or 12, which is consistent with the original theoretical calculation.

If we remove the size factor and just look at the query count, then the DNSSEC-signed zone attracts between 1.4x and 1.6x the queries of the unsigned zone (Figure 4). The theory says that the DNSSEC-signed zone will generate 3 queries from DNSSEC-validating resolvers, so if 20% of clients use DNSSEC-validating resolvers then the authoritative name server will see 1.4x the query load. The rise to 1.6x the query load is consistent with the DNSSEC_signed zone receiving 4x the queries as compared to the unsigned zone. Part of the reason for this discrepancy is a slow rise in the number of duplicate queries we observe at the authoritative name server, where related DNS resolvers ask the server the same query virtually in parallel. This has been seen in from resolvers that appear to be part of a DNS resolver farm, where the original query is sent to multiple resolver engines in parallel as a means of improving robustness and cached performance of the resolver farm.

In this case the effects of caching of the name are not necessarily a factor in these calculations. While the experiment uses unique names that are not found in an DNS resolver's cache, this situation is analogous to a highly used name where individual resolvers periodically refresh their cache from the authoritative name server. What we are interested in here is not the absolute query and byte counts, but the relative ratio of these counts between serving a DNSSEC-signed name and a name that is not DNSSEC-signed.

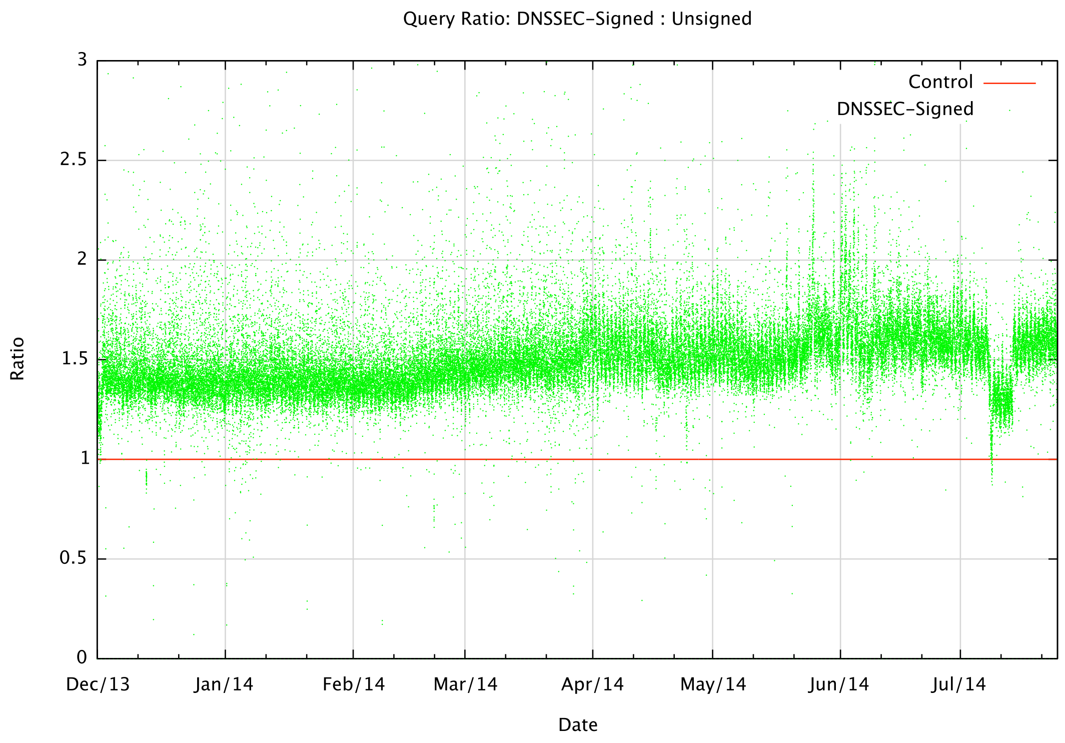


Figure 4 – Query and Response Traffic Ratios: Signed vs Unsigned

Today, DNSSEC-signing a domain name would increase the total query load at the authoritative name servers by a factor of around 1.5x and increase the outbound traffic by a factor of around 8x. We can also make the prediction that if everyone on the Internet were to use DNSSEC-validating resolvers, then signing a name would incur a cost at the authoritative name server of around 4x the query load, and around 12x the traffic load in responses.

Good vs Bad

The results so far compare the DNS resolution of a name that is not signed with that of a DNSSEC-signed name. But, despite the best of intentions, there are times when a DNSSEC signature fails to validate. Now of course there are many reasons why this may happen. The signing keys could expire, or the parent / child key relationship could fail, or the zone could be corrupted in many ways. In addition, the problem need not be exclusively a problem with the zone file and its authoritative name servers. The resolver may not support the signing algorithm, or the root key priming set may be out of date (see "Rollover and Die," (http://www.potaroo.net/ispcol/2010-02/rollover.pdf) for example).

What happens when validation fails?

From the end client's perspective its clear that there is a problem, but its not clear what that problem is. A DNSSEC-aware resolver when encountering a DNSSEC validation problem returns a response to the original query with a response code that, rather cryptically, states that there is a "server fail." The original intended interpretation of this response code was that the name being resolved may well be perfectly fine, but the server who attempted to resolve the name was unable to do so. Now part of the reason why many end clients use multiple DNS resolvers is to respond in this scenario. If the query to one resolver fails then the next step is to retry the query with the other resolver. Even when that query fails with the same error code all it not lost. The query can be repeated with the EDNS0 size option set down to 512. So clients tend to launch repeat queries when using DNSSEC-validating resolvers and when the DNSSEC signature is broken.

It's not just end clients who repeat queries. As the "Roll Over and Die" article illustrates, some, older versions of DNS resolver code walked backward up the name delegation hierarchy and tried to validate the DNSSEC signature using all possible vectors of zone nameservers in the signature query sequence. Only when this exhaustive query failed would the resolver return a Server Failure code.

The impact of this diligence on the part of resolvers to attempt to find a good validation path for the digital signature is illustrated in Figure 5. One quarter of all end clients that perform DNSSEC validation use DNSSEC-validating resolvers that ask queries of a badly-signed DNSSEC name for more than 6 seconds. One sixth of these clients use DNSSEC-validating resolvers that persist in trying to find a good validation path for more than 20 seconds. So the impact on a client using DNSSEC-aware resolvers of a badly signed name is obviously that for DNSSEC-validating resolvers the name cannot be resolved, but for many such clients this bad news takes an extremely long time to arrive.

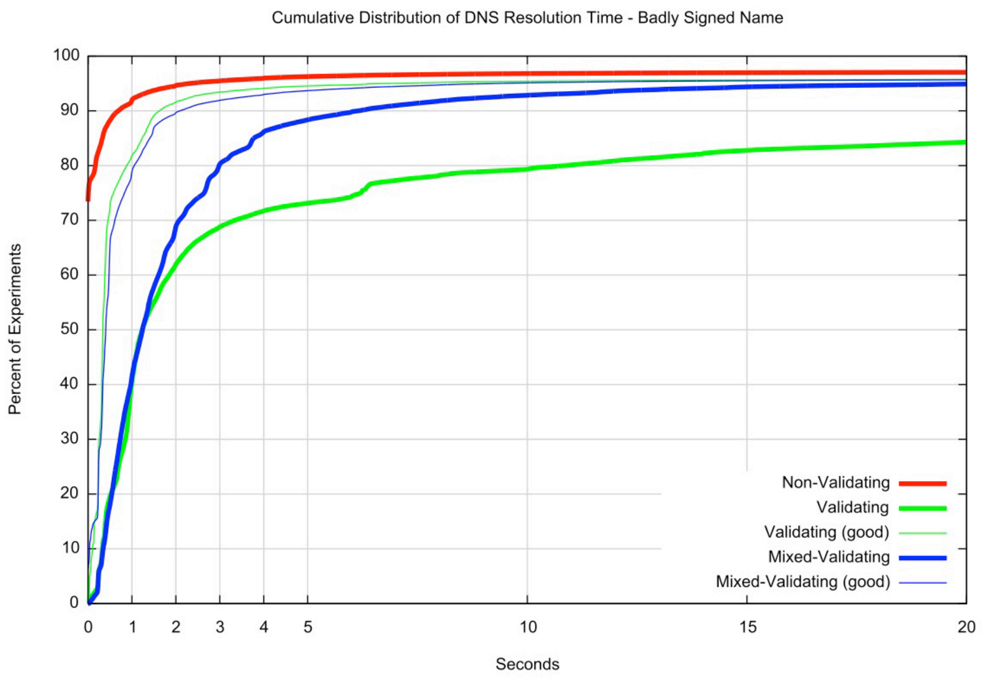


Figure 5 – DNS Query Time for a badly-signed Name

So can we quantify the amplification factor of queries at the authoritative name server when the DNSSEC signature fails? The relative traffic profile associated with serving a properly signed name and a badly signed name is shown in Figure 6. The badly-signed name generates 3x the response traffic as compared to a correctly-signed name. The inference from this is that if every DNS resolver were DNSSEC-aware then the authoritative name server would see an increase of 24x over the traffic load associated with serving an unsigned name. However, the comparison between a good and bad signed names is not quite so large, and the observed impact of a bad signature is some 3x the traffic load as compared to a properly signed name.

However, this is a lower bound rather than a realistic indicator of what to expect. When we examined the ratio of traffic for an unsigned name as compared to a properly signed name in Figure 4, the comment was made that this ratio was not going to be altered if the name was cachable, as long as the cache parameters were the same for both names. What the authoritative name server sees as incoming queries are those queries that have generated a cache miss at the local resolver.

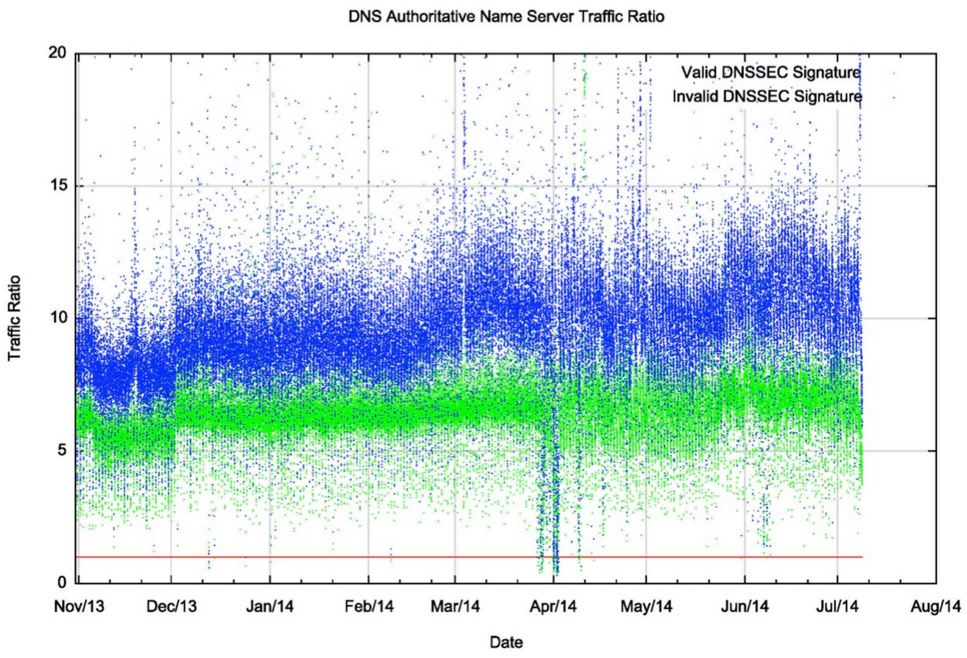


Figure 6 – DNS Response Traffic for a badly-signed Name

However, if the name is badly signed, then DNSSEC-aware resolvers should not cache the response, as the outcome of the name resolution attempt is not trustable. The inference is that this observed relative load factor of 10x only relates to a comparison of the resolution of uncached names. If the name is conventionally cacheable, and the DNSSEC signature fails, then the increase in the query and traffic rate at the authoritative name server is larger, in so far as every DNSSEC-aware resolver will generate a cache miss on each name resolution attempt.

The increase in traffic load as a result of a bad DNSSEC signature depends on the original cache parameters and the original cache hit rate. The increase in relative load at the authoritative nameserver as a consequence of an invalid DNSSEC signature could be arbitrarily large, as the outcome is the product of a basic amplification factor of 3x multiplied by the cache hit rate for an equivalent properly signed name.

The Cost of DNSSEC

The efficient operation of the DNS relies on effective caching, and adding DNSSEC to the mix does not alter this observation. The cost to the client of adding DNSSEC signature to a name is effectively absorbed by DNS caching.

The cost to the name servers is somewhat larger, as the larger DNSSEC responses imply a higher traffic load. Today, signing a name would imply that the query load would increase by 1.5x and the outbound traffic by 8x. If everyone on the Internet were to use DNSSEC-validating resolvers, then signing a name would incur a cost at the authoritative name server of around 4x the query load, and around 12x the traffic load in responses.

It is essential that DNSSEC signatures are properly maintained. If the signatures are bad then the query load is far higher. At a minimum you can expect a query load double that of a properly signed name, and a similar doubling of the traffic volumes. But that's a minimum. It probably will be far higher. And the imposed pain at the server is without any offset gain. When the signature is invalid than DNSSEC-aware resolvers, and their clients, will be unable to resolve the name.

The following are four packet logs, looking at the packets sent and received by a DNS resolver when attempting to resolve particular names. The packet dumps were constructed using the *tcpdump* tool.

Packet Log 1

A query for a 5-label name when the resolver has no cached state, when the resolver is not DNSSEC-aware and where the zone is unsigned:

Firstly, query a root nameserver for the desired name. The response in this case will be the name servers for the .net zone

```
04:11:29.066564 IP6 (hlim 64, next-header UDP (17) payload length: 63)
   2001:388:1000:120:d267:e5ff:feef:a842.52696 > 2001:500:2f::f.53:
   61077% [1au] A? z.00001.z.dashnxdomain.net. (55)
04:11:29.246097 IP6 (hlim 52, next-header UDP (17) payload length: 755)
   2001:500:2f::f.53 > 2001:388:1000:120:d267:e5ff:feef:a842.52696:
   61077- 0/15/16 (747)
```

Secondly, query a .net nameserver for the desired name. The response in this case will be the name servers for the .dashnxdomain.net zone

```
04:11:29.247225 IP6 (hlim 64, next-header UDP (17) payload length: 63)
   2001:388:1000:120:d267:e5ff:feef:a842.65467 > 2001:503:a83e::2:30.53: [udp sum
   ok] 43390% [1au] A? z.00001.z.dashnxdomain.net. (55)
04:11:29.394609 IP6 (hlim 53, next-header UDP (17) payload length: 616)
   2001:503:a83e::2:30.53 > 2001:388:1000:120:d267:e5ff:feef:a842.65467: [udp sum
   ok] 43390- 0/6/3 (608)
```

Thirdly, query a dashnzdomain.net nameserver for the desired name. The response in this case will be the name servers for the z..dashnxdomain.net zone

```
04:11:29.395087 IP (ttl 64, id 6961, flags [none], proto UDP (17), length 83)
    202.158.221.222.60314 > 203.133.248.6.53: 44653% [1au]
    A? z.00001.z.dashnxdomain.net. (55)
04:11:29.414115 IP (ttl 57, id 33300, flags [none], proto UDP (17), length 117)
    203.133.248.6.53 > 202.158.221.222.60314: 44653- 0/1/2 (89)
```

Finally, query a z.dashnzdomain.net nameserver for the desired name. The response in this case will be the desired A resource record:

```
04:11:29.414442 IP (ttl 64, id 6965, flags [none], proto UDP (17), length 83)
    202.158.221.222.55384 > 199.102.79.186.53: 43617% [1au] A?
    z.00001.z.dashnxdomain.net. (55)
04:11:29.602066 IP (ttl 48, id 18278, flags [none], proto UDP (17), length 134)
    199.102.79.186.53 > 202.158.221.222.55384: 43617*- 1/1/2 z.00001.z.dashnxdomain.net.
    A 199.102.79.188 (106)
```

Packet Log 2

Second query, performed immediately following the query in Log 1, changing only the terminal label.

Query a z.dashnzdomain.net nameserver for the desired name. The response in this case will be the desired A resource record:

```
04:12:56.345153 IP (ttl 64, id 7083, flags [none], proto UDP (17), length 83)
    202.158.221.222.51999 > 199.102.79.186.53: 17516% [1au]
    A? w.00001.z.dashnxdomain.net. (55)0
04:12:56.536598 IP (ttl 48, id 33266, flags [none], proto UDP (17), length 134)
    199.102.79.186.53 > 202.158.221.222.51999: 17516*-1/1/2
    w.00001.z.dashnxdomain.net. A 199.102.79.188 (106)
```

Packet Log 3

A query for a 5-label name when the resolver has no cached state, when the resolver is DNSSEC-aware and where the zone is signed:

The first thee queries are similar to the unsigned case, as the resolver "walks" down from the root zone to find a name server than can answer its query.

```
04:17:21.754342 IP6 (flowlabel 0xb1650, hlim 64, next-header UDP (17) payload length: 62)
    2001:388:1000:120:d267:e5ff:feef:a842.59944 > 2001:500:2f::f.53:  17055% [1au]
    A? z.00001.z.dotnxdomain.net. (54)
04:17:21.914905 IP6 (hlim 56, next-header UDP (17) payload length: 754)
    2001:500:2f::f.53 > 2001:388:1000:120:d267:e5ff:feef:a842.59944:  17055- 0/15/16 (746)

04:17:21.916212 IP (ttl 64, id 7264, flags [none], proto UDP (17), length 82)
    202.158.221.222.56998 > 192.33.14.30.53: 19606% [1au] A? z.00001.z.dotnxdomain.net. (54)
04:17:21.922049 IP (ttl 58, id 22212, flags [none], proto UDP (17), length 397)
    192.33.14.30.53 > 202.158.221.222.56998: 19606- 0/5/3 (369)

04:17:21.922450 IP (ttl 64, id 7265, flags [none], proto UDP (17), length 82)
    202.158.221.222.57617 > 203.133.248.6.53: 12845% [1au] A? z.00001.z.dotnxdomain.net. (54)
04:17:21.941541 IP (ttl 57, id 45515, flags [none], proto UDP (17), length 550)
    203.133.248.6.53 > 202.158.221.222.57617: 12845- 0/4/3 (522)
```

This is the query that provides the address. In this case the response is larger because the response includes the RR signature (RRSIG), as does the authority section and the additional records

```
04:17:21.941932 IP (ttl 64, id 7266, flags [none], proto UDP (17), length 82)
    202.158.221.222.61327 > 199.102.79.186.53: 27943% [1au] A? z.00001.z.dotnxdomain.net. (54)
04:17:22.136331 IP (ttl 48, id 16765, flags [none], proto UDP (17), length 1123)
    199.102.79.186.53 > 202.158.221.222.61327: 27943*- 2/4/3 z.00001.z.dotnxdomain.net.
    A 199.102.79.186, z.00001.z.dotnxdomain.net. RRSIG (1095)
```

We now commence the DNSSEC validation phase, by walking back up the name hierarchy querying for DNSKEY and DS records for the A records, but first the process starts with a query of the NS record of this zone, in order to obtain the signature RR of the NS record. AS the local server is sited on a dual stack system it asks for both A and AAAA RR's

```
04:17:22.137317 IP (ttl 64, id 7267, flags [none], proto UDP (17), length 79)
    202.158.221.222.64950 > 199.102.79.186.53: 38854% [1au] A? nsz1.z.dotnxdomain.net. (51)
04:17:22.137485 IP (ttl 64, id 7268, flags [none], proto UDP (17), length 79)
    202.158.221.222.55912 > 199.102.79.186.53: 55544% [1au] AAAA? nsz1.z.dotnxdomain.net. (51)
04:17:22.325473 IP (ttl 48, id 16815, flags [none], proto UDP (17), length 527)
    199.102.79.186.53 > 202.158.221.222.55912: 55544*- 0/4/1 (499)
04:17:22.330897 IP (ttl 48, id 16816, flags [none], proto UDP (17), length 467)
    199.102.79.186.53 > 202.158.221.222.64950: 38854*- 2/2/1 nsz1.z.dotnxdomain.net.
    A 199.102.79.186, nsz1.z.dotnxdomain.net. RRSIG (439)
```

We now ask for the zone signing key value of the terminal zone

```
04:17:22.331245 IP (ttl 64, id 7269, flags [none], proto UDP (17), length 80)
    202.158.221.222.51650 > 199.102.79.186.53: 22395% [1au] DNSKEY? 00001.z.dotnxdomain.net.
    (52)
04:17:22.525114 IP (ttl 48, id 16838, flags [none], proto UDP (17), length 742)
    199.102.79.186.53 > 202.158.221.222.51650: 22395*- 4/0/1 00001.z.dotnxdomain.net.
    DNSKEY, 00001.z.dotnxdomain.net. DNSKEY, 00001.z.dotnxdomain.net.
    RRSIG, 00001.z.dotnxdomain.net. RRSIG (714)
```

At this point Bind9.9 asks the grandparent zone for the Ds record, and receives a null answer, with the parent's zone authority completed, and the addresses of the parent zone name servers

```
04:17:22.525953 IP (ttl 64, id 7270, flags [none], proto UDP (17), length 80)
    202.158.221.222.62192 > 203.133.248.110.53: 50645% [1au] DS? 00001.z.dotnxdomain.net. (52)
04:17:22.545016 IP (ttl 57, id 45537, flags [none], proto UDP (17), length 548)
    203.133.248.110.53 > 202.158.221.222.62192: 50645- 0/4/3 (520)
```

The query is repeated at the parent zone, and a signed response is obtained.

```
04:17:22.545423 IP (ttl 64, id 7271, flags [none], proto UDP (17), length 80)
    202.158.221.222.63384 > 199.102.79.186.53: 25291% [1au] DS? 00001.z.dotnxdomain.net. (52)
04:17:22.733439 IP (ttl 48, id 16868, flags [none], proto UDP (17), length 341)
```

```
    199.102.79.186.53 > 202.158.221.222.63384: 25291*- 3/0/1 00001.z.dotnxdomain.net.
DS, 00001.z.dotnxdomain.net. DS, 00001.z.dotnxdomain.net. RRSIG (313)
```

The query moves up one level and we ask for the DNSKEY zone signing key

```
04:17:22.734114 IP (ttl 64, id 7279, flags [none], proto UDP (17), length 74)
    202.158.221.222.57741 > 199.102.79.186.53: 48709% [1au] DNSKEY? z.dotnxdomain.net. (46)
04:17:22.925279 IP (ttl 48, id 16891, flags [none], proto UDP (17), length 724)
    199.102.79.186.53 > 202.158.221.222.57741: 48709*- 4/0/1 z.dotnxdomain.net.
    DNSKEY, z.dotnxdomain.net. DNSKEY, z.dotnxdomain.net. RRSIG, z.dotnxdomain.net.
    RRSIG (696)
```

Again, Bind 9.9 tries to query the grandparent zone for the record, and it receives a negative response.

```
04:17:22.926187 IP (ttl 64, id 7282, flags [none], proto UDP (17), length 74)
    202.158.221.222.53393 > 192.31.80.30.53: 158% [1au] DS? z.dotnxdomain.net. (46)
04:17:23.127342 IP (ttl 52, id 33537, flags [none], proto UDP (17), length 389)
    192.31.80.30.53 > 202.158.221.222.53393: 158- 0/5/3 (361)
```

The query is repeated at the parent zone, with a signed response

```
04:17:23.127725 IP (ttl 64, id 7283, flags [none], proto UDP (17), length 74)
    202.158.221.222.52792 > 203.133.248.6.53: 45200% [1au] DS? z.dotnxdomain.net. (46)
04:17:23.146763 IP (ttl 57, id 45572, flags [none], proto UDP (17), length 333)
    203.133.248.6.53 > 202.158.221.222.52792: 45200*- 3/0/1 z.dotnxdomain.net.
    DS, z.dotnxdomain.net. DS, z.dotnxdomain.net. RRSIG (305)
```

The query moves up another level to query for the DNSKEY zone signing key.

```
04:17:23.147415 IP (ttl 64, id 7284, flags [none], proto UDP (17), length 72)
    202.158.221.222.49204 > 203.133.248.110.53: 48280% [1au] DNSKEY? dotnxdomain.net. (44)
04:17:23.166491 IP (ttl 57, id 45573, flags [none], proto UDP (17), length 718)
    203.133.248.110.53 > 202.158.221.222.49204: 48280*- 4/0/1 dotnxdomain.net.
    DNSKEY, dotnxdomain.net. DNSKEY, dotnxdomain.net. RRSIG, dotnxdomain.net.
    RRSIG (690)
```

At this point Bind 9.9 moves up two levels to the root zone and queries for the DS RR, and the root zone name servers.

```
04:17:23.167529 IP6 (flowlabel 0x90277, hlim 64, next-header UDP (17) payload length: 52)
  2001:388:1000:120:d267:e5ff:feef:a842.62536 > 2001:500:3::42.53:  26784% [1au]
  DS? dotnxdomain.net. (44)
04:17:23.167623 IP6 (flowlabel 0x9f528, hlim 64, next-header UDP (17) payload length: 36)
  2001:388:1000:120:d267:e5ff:feef:a842.62841 > 2001:500:3::42.53:  21863% [1au] NS? . (28)

04:17:23.175688 IP6 (hlim 60, next-header UDP (17) payload length: 744)
  2001:500:3::42.53 > 2001:388:1000:120:d267:e5ff:feef:a842.62536:  26784- 0/15/16 (736)
04:17:23.175753 IP6 (hlim 60, next-header UDP (17) payload length: 921)
  2001:500:3::42.53 > 2001:388:1000:120:d267:e5ff:feef:a842.62841:  21863*- 14/0/25 .
  NS a.root-servers.net., . NS b.root-servers.net., . NS c.root-servers.net., .
  NS d.root-servers.net., . NS e.root-servers.net., . NS f.root-servers.net., .
  NS g.root-servers.net., . NS h.root-servers.net., . NS i.root-servers.net., .
  NS j.root-servers.net., . NS k.root-servers.net., . NS l.root-servers.net., .
  NS m.root-servers.net., . RRSIG (913)
```

The DS query is repeated to the .net servers

```
04:17:23.176537 IP (ttl 64, id 7285, flags [none], proto UDP (17), length 72)
    202.158.221.222.59876 > 192.35.51.30.53: 63121% [1au] DS? dotnxdomain.net. (44)
04:17:23.345507 IP (ttl 52, id 56065, flags [none], proto UDP (17), length 967)
    192.35.51.30.53 > 202.158.221.222.59876: 63121*- 3/14/16 dotnxdomain.net.
    DS, dotnxdomain.net. DS, dotnxdomain.net. RRSIG (939)
```

The zone signing key for the .net zone is queried

```
04:17:23.346507 IP (ttl 64, id 7286, flags [none], proto UDP (17), length 60)
    202.158.221.222.57105 > 192.5.6.30.53: 26828% [1au] DNSKEY? net. (32)
04:17:23.578772 IP (ttl 51, id 39710, flags [none], proto UDP (17), length 771)
    192.5.6.30.53 > 202.158.221.222.57105: 26828*- 3/0/1 net. DNSKEY, net. DNSKEY, net.
    RRSIG (743)
```

The DS record for .net is retrieved from the root zone

```
04:17:23.579460 IP (ttl 64, id 7287, flags [none], proto UDP (17), length 60)
```

```
    202.158.221.222.50844 > 202.12.27.33.53: 40321% [1au] DS? net. (32)
04:17:23.855605 IP (ttl 51, id 36436, flags [none], proto UDP (17), length 267)
    202.12.27.33.53 > 202.158.221.222.50844: 40321*- 2/0/1 net. DS, net. RRSIG (239)
```

Packet Log 4

Second query, immediately following Log 3, changing only the terminal label.

```
08:08:31.984563 IP (ttl 64, id 12492, flags [none], proto UDP (17), length 82)
    202.158.221.222.64377 > 199.102.79.186.53: 39737% [1au] A?
    x.00001.z.dotnxdomain.net. (54)
08:08:32.175677 IP (ttl 48, id 46947, flags [none], proto UDP (17), length 1123)
    199.102.79.186.53 > 202.158.221.222.64377: 39737*- 2/4/3
    x.00001.z.dotnxdomain.net. A 199.102.79.186, x.00001.z.dotnxdomain.net.
    RRSIG (1095)
```

Disclaimer

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