The ISP Column
A monthly column on things Internet

May 2013
Geoff Huston
George Michaelson

Measuring DNSSEC Performance

In February and March of 2013 we conducted a set of measurement tests to understand the extent to which today's Internet-connected end systems are capable of performing validation of DNS responses.

The experiment tested some 5.3 million end systems, using test code within an advertisement to perform the test, and using Google's online ad delivery system to deliver the test to end systems. Of these 5.3 million end systems that received the impression of the advertisement, some 4 million completed the test code, and of these systems some 120,000 systems, or 3% of the tested set appeared to use exclusively DNS resolvers that performed DNSSEC validation. In other words it appears that some 3% of end systems are "covered" by DNSSEC. A further 2% of end systems appear use a combination of DNSSEC-validating and non-validating resolvers. The remaining systems do not generate any DNSSEC queries at all when resolving DNSSEC-signed DNS names.

There have been a number of reasons why both domain name administrators and vendors of client DNS software cite for not incorporating DNSSEC signing. The added complexity of the name administration process when signatures are added to the mix, the challenges of maintaining current root trust keys, and the adverse consequences of DNSSEC signature validation failure have all been mentioned as reasons to hesitate. We have also heard concerns over increased overhead of using DNSSEC. These concerns come from zone administrators, authoritative name server operators and from suppliers of DNS resolver systems, and all point to a concern over the imposition of further overheads in the process of DNS name resolution when the name being resolved is DNSSEC signed. While the issues of complexity are challenging to quantify, we were interested in the issues of performance. What are the performance costs of adding DNSSEC signatures to a domain?

In measuring the "performance" of DNSSEC there are a number of possible considerations we'd like to address:

- In having DNSSEC-validating DNS resolvers assemble a signature validation chain for the signed resource record, what is the incremental time for resolution of DNS queries if the end system uses a DNSSEC-validating resolver?

- In serving a DNSSEC-signed zone do we see a considerable increase in the number of queries made to the zone's authoritative name server? Can we quantify the increased server query load?

- In providing signatures to DNSSEC responses what is the extent of increased DNS traffic due to the larger DNS response packets?

- What's the performance implications of an invalidly-signed DNSSEC signature?

This article will explore these performance questions, comparing the predictions from the model of DNSSEC operation to the observed behaviour of the end systems that were tested for DNSSEC support.

The DNSSEC Measurement

Firstly, a recap of the measurement exercise we use to test DNSSEC use.

The experiment uses an online advertisement campaign to deliver the test code to end systems. When the end system is passed an ad that is carrying the experiment the system runs embedded Adobe Flash code. The code is executed when the ad is passed to the user, and does not rely on a user "click" or any other user trigger action. The active code interrogates one of two experiment controllers by performing a URL fetch. The contents of the fetched experiment control URL are a dynamically generated sequence of four URLs. These four URLs are the substance of the test setup

- The first URL is that of a transparent 1x1 png image file. This URL uses a domain name that has been DNSSEC-signed, and the DNSSEC signature is valid, in that there exists a valid sequence of DNSKEY and DS records from the DNS root zone to this URL. This is the **A** test:
  `http://a.u7280280162.s1364784185.v6022.69da1.z.dotnxdomain.net/1x1.png`

- The second URL is also a transparent 1x1 png image file. The URL uses a domain name that is similarly structures to the "d" URL, but where the domain name is not DNSSEC-signed. This is the **B** test:
  `http://b.u7280280162.s1364784185.v6022.69da1.z.dashnxdomain.net/1x1.png`

- The third URL is also a transparent 1x1 png image file. The URL uses a domain name that is similarly structured to the **A** URL, but where the domain name is DNSSEC-signed. The difference between this domain name and the first domain name is that here the DNSSEC validation is configured to fail, as the validation path is deliberately broken. In this case the breakage is by a deliberate break in the signature validation chain, where a DS Resource record does not contain the hash of any of the corresponding DNSKEY keys. This is the **C** test:
  `http://c.u7280280162.s1364784185.v6022.69da2.z.dotnxdomain.net/1x1.png`

- The fourth and last URL is used by the end system to pass data back to the experiment server. The DNS name used in this URL is not signed. The parameters added to the URL when the object is fetch is the times, as measured by the end system, to fetch each of the first three URLs. This URL is triggered to be fetched by the end system when either all three (**A**, **B** and **C**) URLs have all been fetched, or a local 10 second timer has elapsed. The purpose of this URL is to allow us to distinguish between experiments where the code runs to completion (or at least for 10 seconds), and where the execution of the test code is aborted by the user. This is the "result" URL:
  `http://results.u7280280162.s1364784185.v6022.69da1.x.rand.apnic.net/1x1.png`

The experiment uses a consistent ordering in the way in which the URLs are loaded so that the end system browser's Adobe Flash subsystem is instructed to load the **A**, **B** and **C** URLs to the local fetch queue in that order.

All four URLs share a unique identifier within the DNS part of the URL. In the above example, it's the string "`u7280280162.s1364784185.v6022`" This allows us to link up the logs of the separate DNS queries and URL fetches into a consistent picture of the execution of each experiment. All four parts of the experiment contain this common string.

The experiment uses some 750,000 unique domain names. Some 500,000 subdomains are located as subdomains of "`z.dotnxdomain.net`". A further 250,000 subdomains are placed under "`z.dashnxdomain.net`". Each of these subdomains contains a single wildcard entry, with a single A RR entry. In the "`dotnxdomain.net`" domain all these subdomains are DNSSEC signed. For odd-numbered subdomains the DNSSEC signature is valid, while in the even numbered subdomains the DNSSEC signature is invalid. The "`dashnxdomain.net`"domain does not use DNSSEC signatures.

While it is feasible to generate a DNS zone file with millions of signed zones, we decided to use an approach that did not require the generation of extremely large DNS zone files. In this experiment we have set the DNS TTL at one hour, and we want to ensure that within a minimum of two hours no unique DNS label is reused. To achieve this we used 250,000 unique labels in the experiment with the one hour TTL, ensuring that as long as the DNS resolvers honour the TTL of cached data we will avoid the issue of DNS cache behaviour masking the DNSSEC validation queries we are looking for.

The rationale for picking these particular DNS size parameters is as follows. We started with examining the Google's ad delivery rate. As every ad impression makes a request from the advertisement control server it is possible to track the number of ad impressions per second. When looking at the number of ad impressions per second on a day-by-day basis, the results are shown in Figure 1.

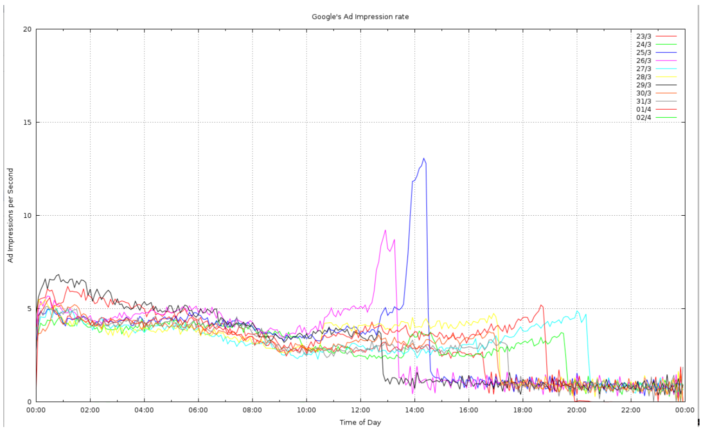

Figure 1 – Per-second Ad Impression Rate

It appears that the Google Ad delivery system normally delivers up to 10 ads per second with the settings we used for the advertising campaign.

But in our case it's not the peak per second rate that is the issue. We are interested in the number of ads that are delivered in any two hour interval. The number of ads delivered in a two hour sliding window on a second-by-second basis for a number of days of an ad campaign is shown in Figure 2.

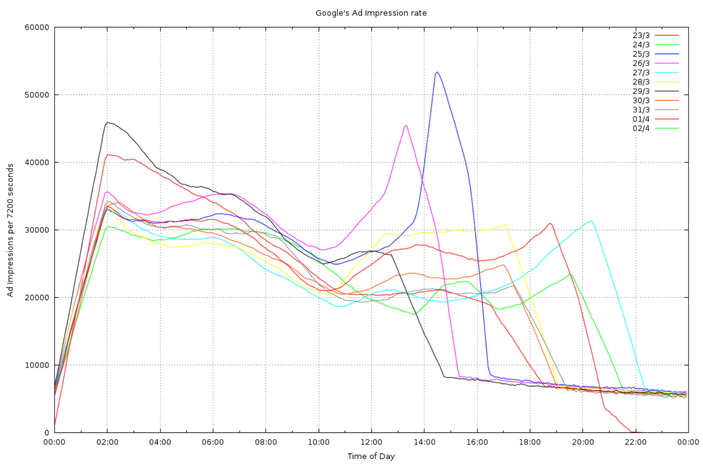

Figure 2 – 2 Hour Sliding Window Per-second Ad Impression Rate

The highest 2 hour period of ad delivery was experienced on the first day of the experiment, where a peak of some 75,000 ads were delivered in a two hour period. The overall result was that in each 24 hour period the ad was delivered between 230,000 and 330,000 times.

> Therefore, an ad delivery system that cycles through 250,000 unique labels will have a cycle period of between 18 to 26 hours. This appears to be adequately longer than the 1 hour TTL on the DNS data.

The experiment is measured at two points:

- The end system's browser measures the elapsed time from the time the fetch URL is dynamically added to the end system's document until the time when the object fetch has successfully completed. In the context of this experiment, this would conventionally generate three time values, each of which measure the time taken to resolve the URL and then undertake an HTTP fetch of the referenced object.

- The server used for this experiment is both the DNS authoritative name server for these domain names and the web server for the URLs. All transactions on this server are logged at both the application level, where the DNS name daemon logs DNS queries as they are received and the web server logs URL fetches as they are completed, and the packet level, where all incoming and outgoings packets are recorded.

This allows each URL fetch to be timed by the end system and by the server.

The Theory of DNSSEC Performance

If we look at the queries made by a end system who uses a non-validating DNS resolver, the sequence is relatively straightforward:

```
DNS  Query -EDC IN A b.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
HTTP GET /crossdomain.xml b.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
HTTP GET /1x1.png b.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
```

The end system makes an initial request for the address of the given URL. Because the URL uses a unique label we can ensure that the DNS subsystem cannot use an already cached response, and the query is passed to our instrumented authoritative name server.

When the end system receives the DNS response, it then proceeds to load the image. As the end system is running the Adobe Flash player, the end system's Flash Player needs to obtain permission to talk to servers other than the one that provided the original Flash code. We see this permission check as a fetch for a `crossdomain.xml` object that contains the policy statements for this server. Once this fetch completes, the end system then fetches the image URL.

How long should these three network transactions take?

The DNS component of this fetch should be no faster than one round trip time interval (RTT) between the end system and the server, and, depending on the level to which the parent domain's details are already cached by the end system's DNS resolver, this may involve additional queries. Because the object contains a unique DNS label we can be confident that the name does not already exist in the DNS resolver's cache, so the end system's query to its DNS resolver will end up in a query being received at our authoritative name server.

The HTTP fetch involves a TCP handshake then a HTTP fetch, and this sequence operates at a minimum of 2 x RTTs. There is no visible evidence of pipelining the two HTTP requests in a single TCP session from our collected data, so this 2 x RTT interval will apply to each HTTP fetch.

This implies that this entire transaction to fetch a web object will take, at a minimum, a 5 x RTT interval, as measured by the end system.

If the domain name is DNSSEC-signed, and the DNS resolver is performing DNSSEC validation, then the sequence of transactions as seen at the experiment's server now includes additional DNS queries related to DNSSEC validation:

```
DNS   Query: -ED IN A a.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
DNS   Query: -ED IN DNSKEY 5e4e3.z.dotnxdomain.net
DNS   Query: -ED IN DS 5e4e3.z.dotnxdomain.net
HTTP GET /crossdomain.xml a.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
HTTP GET /1x1.png a.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
```

Again, at a minimum, this sequence of transactions will take 7 x RTTs, with the additional time being used to perform the DNSKEY and DS queries.

The expectation here is that if the end system uses DNSSEC-validating DNS resolvers then the elapsed time to fetch the **A** URL (DNSSEC-signed) should take approximately 1.5 times the amount of time taken to fetch the **B** (unsigned) URL.

The **C** URL, with the invalid DNSSEC signature, presents an interesting case. If the end system is using a DNSSEC-validating DNS resolver then the DNS query will generate a DNS response with a "Server Fail" error code. This DNSSEC failure would conventionally cause the end system to repeat the query to any other configured DNS resolvers. If this second resolver is not a DNSSEC-validating end system, then the query will generate a response with the requested address.

```
DNS   Query: -ED IN A c.u94278337.s1364428957.v6022.5e4e4.z.dotnxdomain.net
DNS   Query: -ED IN DNSKEY 5e4e4.z.dotnxdomain.net
DNS   Query: -ED IN DS 5e4e4.z.dotnxdomain.net
DNS   Query -EDC IN A c.u94278337.s1364428957.v6022.5e4e4.z.dashnxdomain.net
HTTP GET /crossdomain.xml c.u94278337.s1364428957.v6022.5e4e4.z.dashnxdomain.net
HTTP GET /1x1.png c.u94278337.s1364428957.v6022.5e4e4.z.dashnxdomain.net
```

So we would expect that the **C** URL will take a minimum of twice as long to fetch than the **B** URL if the client is performing DNSSEC validation on only some of its configured DNS resolvers. On the other hand, if all the client's DNS resolvers are performing DNSSEC validation then there will be no fetch of the object.

Measuring DNSSEC Performance

DNS Name Resolution Performance and DNSSEC

The first question to look at is:

> "In having DNSSEC-validating DNS resolvers assemble a signature validation chain for the signed resource record, what is the incremental time for resolution of DNS queries if the end system uses a DNSSEC-validating resolver?"

The experiment has produced a set of client-side times for the retrieval of the three objects. For each experiment we can classify the client as either a client that exclusively uses DNSSEC-validating resolvers, a client that uses a mix of DNSSEC-validating resolvers, or a client that does not use DNSSEC validating resolvers. If we compare the client-side time to load URL B (unsigned domain name) with URL A (validly DNSSEC-signed domain name), then, as shown above we expect to see that A will take 1.5 times as long to load as B for clients who use DNSSEC-validing resolvers, and for A to load in precisely the same time as B if the resolvers are not performing DNSSEC validation. The result of this experiment which shows the spread of client-measured times for the time to fetch URL a minus the time to fetch URL B is shown in Figure 3.

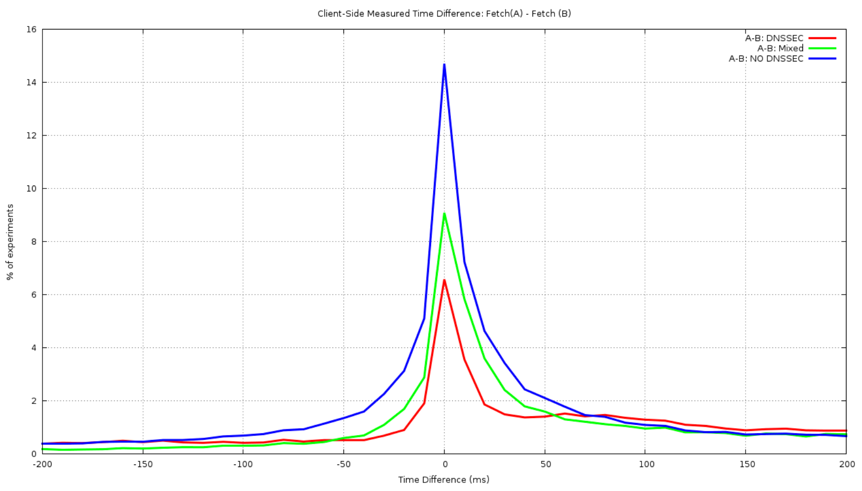

Figure 3 – Client-Side timer Fetch(B) – Fetch(A)

This is highly surprising result. We had expected a result as shown in Figure 4, where the time taken to fetch A and B were precisely the same in the non-DNSSEC case, and exclusively longer when the client used DNSSEC-validating resolvers.

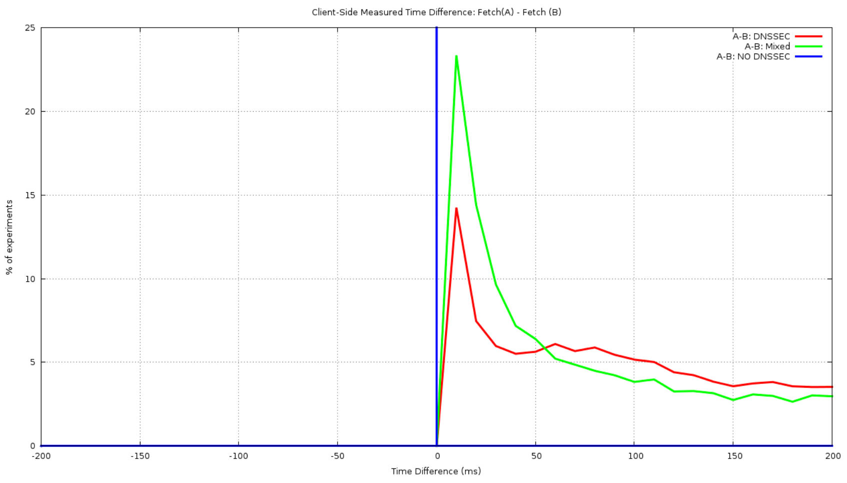

Figure 4 – Client-Side timer Fetch(B) – Fetch(A) – Theoretical Prediction

Our assumption is that the browser is functioning in a threaded mode, where the fetch of these URLs proceeds in parallel, and the operation of fetching one object does not impede the fetch of the other objects. However, for this experiment, the fetches are being performed by the user's Flash engine, and the experimental results strongly indicate that this assumption of parallel operation by the user's Flash engine is a flawed assumption.

There is a pointer in the slightly higher proportion of clients that take longer to fetch A in Figure 3 to conclude that some clients take longer to perform DNSSEC validation, but it is not a clear outcome from the client side timings.

If the use of a Flash engine by the client renders client-side timings in this experiment an unreliable indicator of performance, are there server-side measurements that could provide a similar view of the difference in fetch times that are imposed by the use of DNSSEC?

What does the server see in terms of the number of DNS queries to resolve each domain name. Figure 5 shows the number of queries made by clients who exclusively use DNSSEC-aware resolvers, and compares that to the number of queries used to resolve the non-DNSSEC-signed experiment.

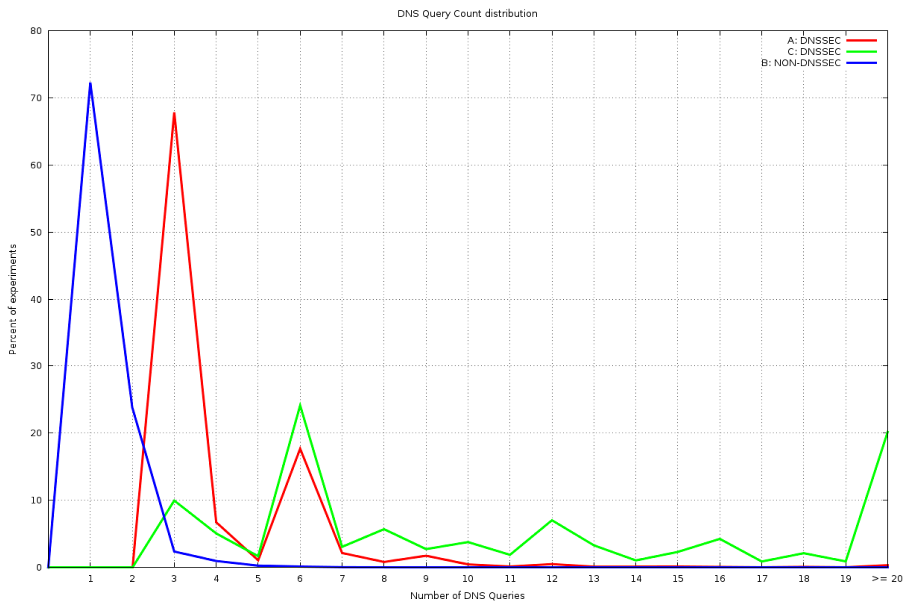

Figure 5 – DNS query count per domain name

Slightly over 70% of the resolutions for the non-DNSSEC-signed domain name, B, complete with a single query, and 95% of resolutions are completed with two queries.

The DNSSEC-signed domain names require a minimum of three queries (A, DNSKEY and DS queries). For the validly-signed domain name 68% of all clients complete their DNS resolution within the three queries. A further 18% of clients perform the resolution function in 6 queries. It is assumed that in the majority of cases of 6 queries the original client query has timed out, and the client has launched a second query before the first query has returned a result.

The invalidly-signed domain shows three local peaks. Some 10% of clients perform 3 queries. Some 22% perform 6 queries. However, some 20% of DNS resolution operations for the invalidly-signed domain generate 20 or more queries. Clearly the consequences of an invalid DNSSEC signature is one that has the potential to significantly increase the DNS query load on the server.

We can look at the elapsed time to complete the DNS query sequence for each experiment, taking the server-side elapsed time from the receipt of the first query for each domain name to the final query, including in this case A, DNSKEY and DS RR queries. This distribution is shown in Figure 6.

The blue line is the query time for the unsigned domain name (B). 30% of the resolutions for the B URL take multiple A queries. Local peaks are visible at 1, 2, 4 and 8 seconds, indicating that some resolvers are using some form of exponential backoff on re-queries.

The red line is the A URL, where it takes three queries to complete the name resolution, and the green line is the invalid DNSSEC domain. While 2% of all resolutions for the A domain name took longer than 10 seconds, 17% of all resolutions for the C domain name (invalid DNSSEC signature) took longer than 10 seconds.

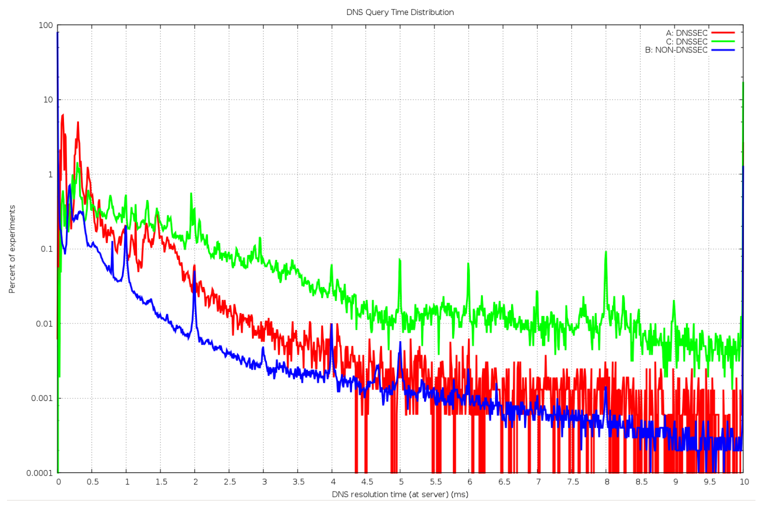

Figure 6 – DNS Resolution Time Distribution

An expansion of the initial 0.5 seconds in Figure 7 shows the effect of DNSSEC in terms of resolution time. There are local peaks at for the A domain name at 150ms, 300ms and 450ms, showing the effects of the increased DNS query count on the total resolution time.

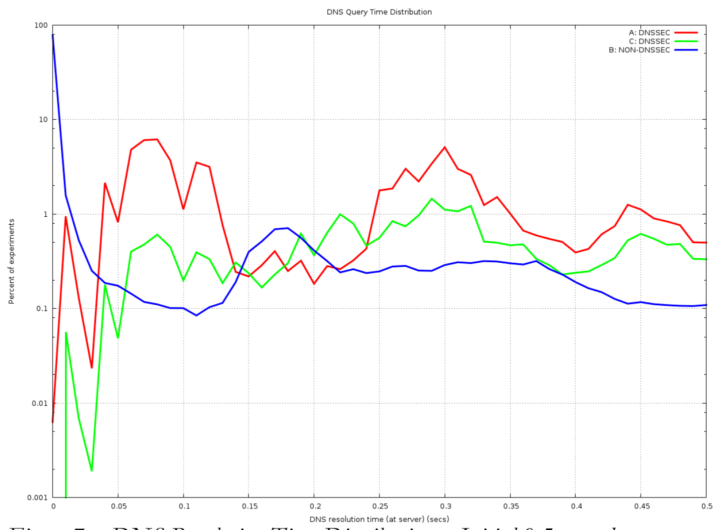

Figure 7 – DNS Resolution Time Distribution – Initial 0.5 seconds

However, the question we are trying to answer here is to measure the increased time to resolve a DNS name when there is the additional factor of DNSSEC validation. Figure 8 shows the cumulative distribution of resolution times for the three domain names.

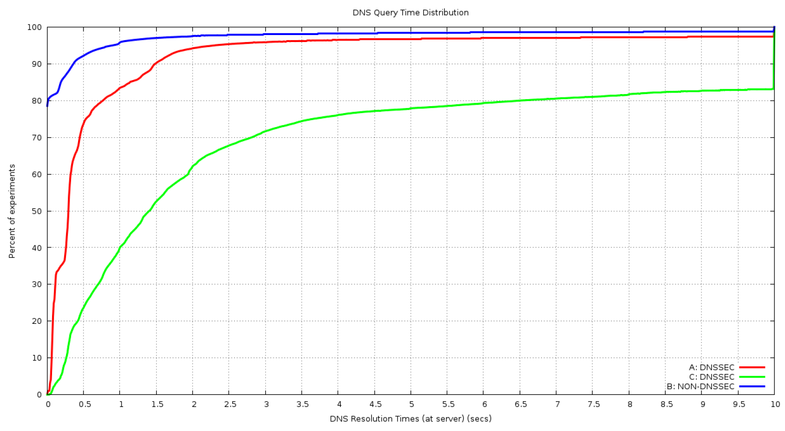

Figure 8 – DNS Resolution Time Distribution – Cumulative Distribution

The green line shows the additional time takes to resolve the DNS name of the C URL. For some 16% of clients that use DNSSEC-validating resolvers the name resolution time took more than 10 seconds. One half of all DNS resolution attempts by these clients take an additional 1.5 seconds.

The red line in Figure 8 shows the additional time taken by clients who use DNSSEC-validating resolvers. Some 10% of clients take 1.5 seconds longer than the single query time, and 25% of clients take an additional 0.5 seconds or longer. The detail of the first 0.5 seconds is show in Figure 9.

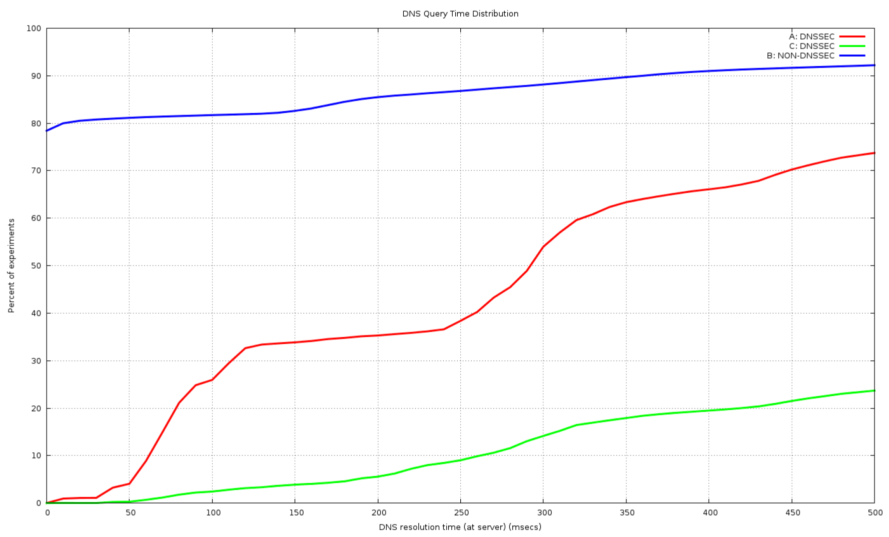

Figure 9 – DNS Resolution Time Distribution – Cumulative Distribution for Initial 0.5 seconds

Some 30% of clients who use DNSSEC-validating resolvers complete the DNS query set within an additional 100ms over the time taken for a single A query, and 50% of this set of clients complete the DNS query operation in 300ms as shown in the red line in Figure 9.

The distribution for the B domain name (blue line) is also worthy of comment. It is noted that 20% of clients fetching the B URL generate multiple queries, which is an unexpected form of outcome until you factor in the clients who are probing for IPv6 AAAA RRs. But even then it is noted that some 8% of clients generate multiple DNS queries that take longer than 500ms to resolve this domain name. This is an unexpectedly high result.

There is something less than healthy about an Internet where a some 8% of clients take more than half a second to perform a very simple query of an uncached domain name!

This figure also shows us that 80% of clients resolve and DNSSEC-validate the A DNS name within an additional 750ms over the time taken to perform a single DNS query. The median incremental time spent to resolve a DNS name by a client who uses DNSSEC-validating resolvers is 280ms when the DNSSEC-signature is valid.

So in a world of a "million dollar millisecond" it may appear in the first instance that as imposition of some 280 milliseconds additional DNS delay is simply too high an imposition. However, there is something unusual about this experiment, in that we have negated the effects of DNS resolver caching. Every fetch of a domain name is forced to also fetch the DNSKEY and DS RRs. Conventionally, we would expect these RR's to be cached and not fetched upon each and every use.

Given that these RR's apply to the zone file, the likelihood of a cache hit is generally far higher than the likelihood of a cache hit on a terminal name. This 280ms average time penalty for DNS name resolution is only applicable for the "first use" cases where the terminal name, and the zone's DNSKEY and DS records are uncached. When the name is held in the cache, the DNS resolution function would be confidently anticipated to occur in far shorter time intervals.

The median time for DNS resolution of the C domain name with an invalid DNSSEC signature is a significantly longer time penalty of 1.4 seconds, and 80% of all clients complete the name resolution of the C URL within 6.5 seconds. There is an issue here, and it relates to the high performance penalty that is incurred when a domain name is signed, but when DNSSEC validation cannot validate the signature. What appears to be the case here is that when validation failure generates a "Server Fail" outcome, the name resolution process then starts a process of launching the query to other DNS resolvers to see if they can resolve the name. In this case the search for a successful resolution can take a significant amount of time, and some 16% of clients took longer than 10 seconds to perform repeated queries to the authoritative name server to resolve the name.

In this case this shows the incremental time penalty of what is perhaps a "best" form of failure. The number of name servers at each zone level has been kept to a single name server for all but the second level domain zone, where two name servers are used. Related work on DNSSEC invalidity ("Roll Over and Die?", February 2010, http://www.potaroo.net/ispcol/2010-02/rollover.html) shows that some resolvers perform an exhaustive test of all authoritative name servers in the event of validation failure, and the more name servers placed on a domain, the larger the query load in the event of validation failure (such as an expired key or similar). In this case caching is of marginal help, as DNSSEC-aware DNS resolvers should reasonably be expected not to cache DNS names that fail DNSSEC validation. In this case our use of a single authoritative name server for the invalid domain produces a DNS resolution result for the client in a shorter time than if we had used multiple name authoritative servers for the domain.

DNS Authoritative Server Performance and DNSSEC

Now let's look at the server's performance. The question we had posed was:

> "In serving a DNSSEC-signed zone do we see a considerable increase in the number of queries made to the zone's authoritative name server? Can we quantify the increased server query load?"

It must be noted here that we have deliberately structured this experiment in a manner that negates caching of answers, so the results presented here are results where the effects of caching are not included.

The query load at the authoritative name server is shown in Figure 10. This load is broken down into three categories: the queries for the A, B and C DNS names. This includes all RR query types, including A and AAA queries (even though there is no AAAA RR used in this experiment). In the case of the A and C DNS names the query load includes the fetches for the DNSKEY and DS records. The absolute query load in Figure 10 is an artefact of the Ad impression rate. However, it must be remembered that every ad includes one unique name in the A, B and C categories, so if all three names were not DNSSEC-signed we would expect all three query traces to show equal query volumes.

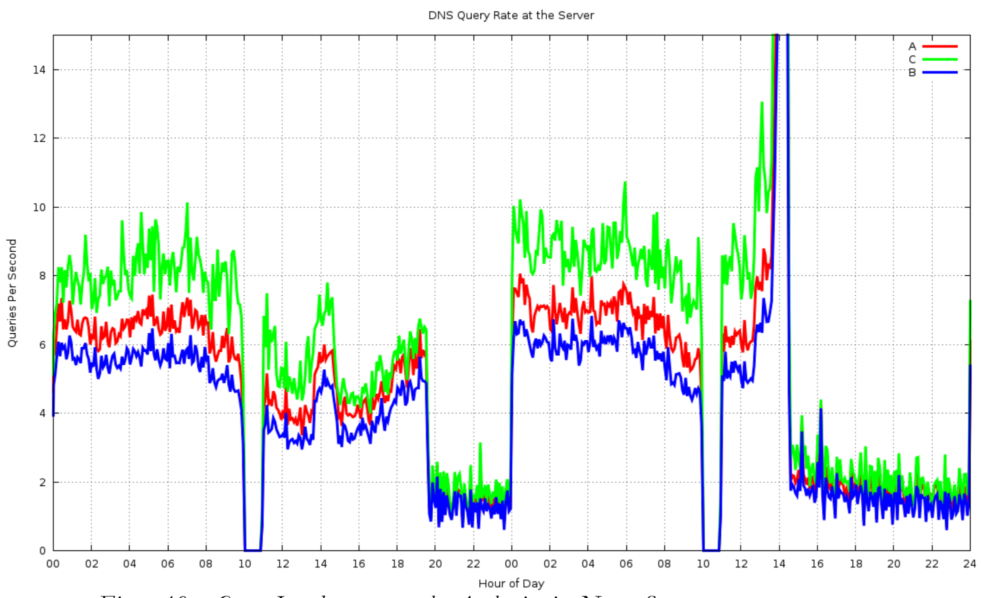

Figure 10 – Query Load as seen at the Authoritative Name Server

Evidently the A query load is visibly higher than the B query load, and the C query load is even higher.

The average over the entire collected data was that the queries for the A URL was 12% higher than the B URL. Some 96% of tests use resolvers that only fetch A RRs (and some also perform a AAAA query as well). The other 4% of tests are also fetching DNSKEY and DS RRs, presumably as part of some form of DNSSEC validation. The minimal case here is that this DNSSEC validation function will entail an additional 2 queries, so that the total query load should rise by 8%. However, as noted above, there is a significant proportion of clients who are using resolvers that are generating more than 3 queries to resolve the A URL, which would be consistent with the observation of a 12% greater query rate at the server for the A domain name. The distribution of the number of DNSSEC RR queries for the A test is shown in Table 1.

Number of DNSSEC RR queries	Number of Tests
1	260
2	116,137
3	836
4	22,547
5	356
6	2,493
7	118
8	5,084
9	884
10	418
11	79
12	403
13	44
14	123
15<=x<100	752
100<=x<1,000	276
1,000<=x	4

Table 1 – DNSSEC RR Query Distribution for "A"

As expected there are a large number of tests with 2 queries, and also peaks at 4, 6, and *, which appear to be repeated queries for the DNSKEY and DS pair from different DNS resolvers. Presumably the use of these additional resolvers is due to client-side timeouts of the original query, where the client poses the query to its alternate DNS resolvers when the initial query does not elicit a response in time.

There are a small number of cases with very large query counts. This appears to point to some form of unresolved operational bug in certain DNS name resolvers.

It would appear that an authoritative name server for a DNSSEC-signed name could expect to see its query rate rise by 12% for the signed domain. But again it must be remembered that this is a worst case scenario, in that the names used in this experiment are uncacheable. The DNSSEC RRs refer to the zone, not the terminal name, so that a cached value of these RRs could be used for all domain names in a signed zone. So it's perhaps more accurate to state that, at present, the increased query rate arising from a DNSSEC-signed zone is estimated to be up to 12%. Obviously if the number of clients using DNSSEC-validating resolvers rises, this figure will rise as well. If the entire set of clients and resolvers were performing DNSSEC validation then the number would be up to 300%, given the current behaviour of DNSSEC-validating DNS resolvers and the observed pattern of client-side behaviours.

What happens when the domain name is not validly signed?

The consequences of invalidity in DNSSEC validation are very serious. In this case these same 4% of clients who are performing some form of validation function for the C (invalid-signed) URL caused the query rate at the server to rise by an average of 37%.

What appears to be happening here is that the resolvers return a "Server Fail" code to the client, which causes the client to re-try the query. The average number of additional queries at the authoritative name server in the case of a domain name with an invalid DNSSEC signature is an average of an additional 9 queries. The distribution of the number of queries for the DNSKEY and DS RRs for the C domain name is shown in Table 2.

Number of DNSSEC RR Queries	Number of Tests
1	275
2	52,177
3	5,775
4	38,986
5	589
6	5,451
7	6,110
8	9,697
9	1,425
10	3,537
11	204
12	4,467
13	204
14	6,717
15	210
16	2,151
17	147
18	1,056
19	1,361
20	996
21	2,063
22	604
23	121
24	882
25	123
26	1,439
27	234
28	1,042
29	121
30<=x<100	3,260
100<=x<=1,000	40
1000<x	0

Table 2 – DNSSEC RR Query Distribution for "C"

In this case the use of additional resolvers is presumably due to the client re-issuing the query to its alternate resolvers upon receipt of the "Server Fail" response from DNSSEC-validating resolvers that are unable to complete validation.

This 37% increase in the query load is generated by 4% of clients who are performing some form of DNSSEC validation. At this rate it the entire client base of the Internet was using DNSSEC-validating resolvers then the increase in query load at the authoritative server caused by an invalid DNSSEC signature would be some 900% . The effects that caching would have on this figure is uncertain, and the nature of the failure would have some bearing on this. This domain also has a single NS record and a single authoritative name server. Adding further authoritative name servers would be anticipated to increase the query load.

There is a current (April 2013) proposal by the operators of the root zone of the DNS to roll the key-signing-key of the root of the DNS. The consequences of this key change would be that DNS resolvers that are not running with up-to-date root keys, or have not upgraded their software to support RFC5011 mechanisms for trust key rollover, will then find that all DNSSEC validation operations that depend on the root key will then fail. This scenario has not been modelled in this particular experiment, but there is some legitimate cause for consternation with this proposal. Not only will this cause all signed domain names to fail DNSSEC validation for all clients who use these validating resolvers who continue to use stale root key information as their trust key set, but the other consequence of such a failure is the potential to generate a significant increase in the query load on the authoritative name servers for the DNSSEC-signed zones if these DNS resolvers perform an exhaustive hunt for a valid DNSSEC key validation path to the root for every query.

The related DNSSEC authoritative name server performance question related to the traffic load imposed on the server:

> "In providing signatures to DNSSEC responses what is the extent of increased DNS traffic due to the larger DNS response packets?"

In this experiment we've been able to isolate the traffic associated with queries and responses for each of the three domain names. They are shown in Figures 11 and 12.

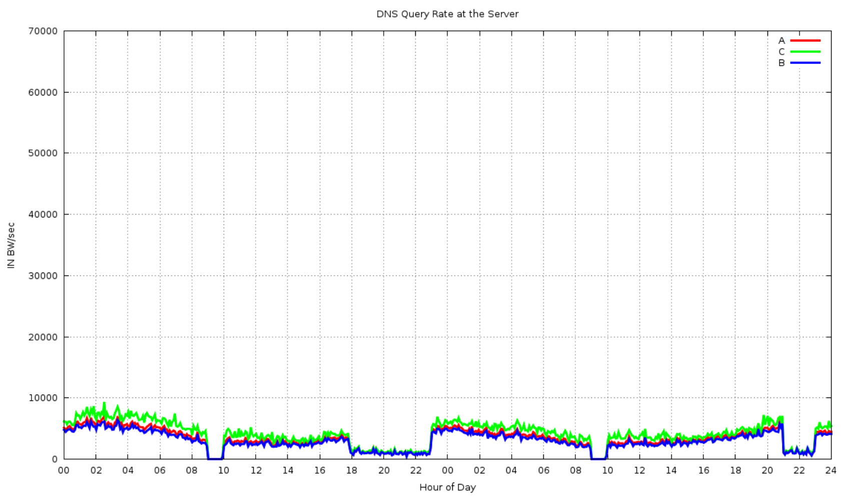

Figure 11 – Input (Query) Traffic Load as seen at the Authoritative Name Server

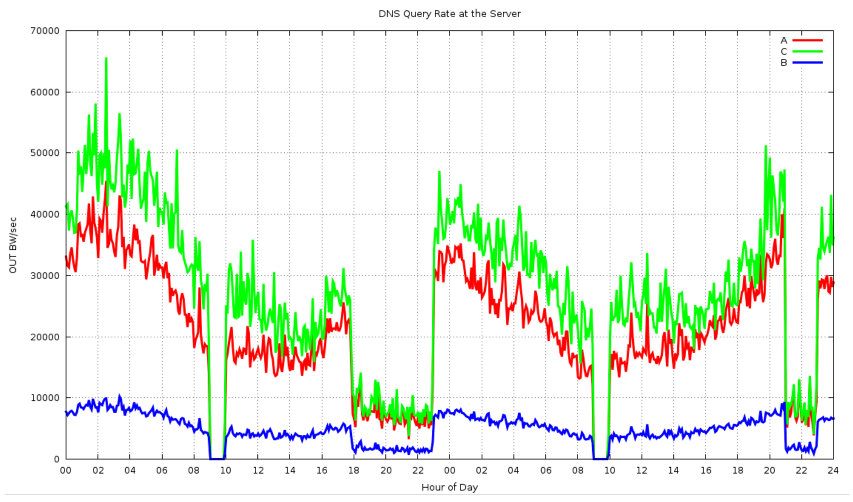

Figure 12 – Output (Response) Traffic Load as seen at the Authoritative Name Server

The traffic for the B domain name shows that the output (response) data rate is some 60% greater than the input (query) data rate, corresponding the average packet sizes of the UDP DNS query and response packets.

For the A DNS name there are 12% more queries, and 16% of the packets now contain DNSSEC signature information. The A response with signatures has grown from some 150 bytes in size to 923 bytes. The DNSKEY RR response is 723 bytes and the DS RR response is 313 bytes. These significantly larger packets increase the traffic load of responses by a factor of 4.3.

This is a significant increase, and perhaps it's worth a little more analysis. The unsigned response is 150 bytes in size, while the signed response of the A resource record, plus the size of the DNSKEY and DS responses is a total of some 1,950 bytes. The inference is that if all clients used DNSSEC-validating resolvers, the traffic levels of an authoritative name server for a DNSSEC-signed domain would increase by up to 13 times if all clients queried efficiently. But as only 4% of clients are performing DNSSEC validation of some form, then we should expect to see some 4% of clients increase their response traffic by some 13 times. That would account for a rise in response traffic volumes by 52%, not by 330%. What we observe is a large set of clients are using resolvers that generate queries with the DNSSEC OK bit set in the query, but do not follow up with any DNSSEC validation queries. For the A URL we observe that 70% of all A queries have the DNSSEC OK bit set. Given that the response to a DNSSEC OK query is 923 bytes, or 6 times the size of the query, then if 70% of all queries elicit this larger response then this would account for the a traffic increase factor of 4.2, while the additional responses for the DNSKEY and DS queries in 4% of cases would account for the additional traffic increase.

So the overall result is that if you DNSSEC sign a domain today then some 70% of the received A queries will request DNSSEC additional information, and the traffic level in responses will rise by a factor of 4.5 over traffic levels for an unsigned domain. If every client used DNSSEC validating resolvers then the total traffic levels would increase by a factor of up to 13 over levels associated with an unsigned domain. Obviously, once more, caching of the DNSSEC zone values would have some impact on this number, and a more accurate working projection is that traffic volumes would increase by a factor of between 6 and 13, depending on the zone's key lifetime and query activity.

For the invalidly-signed domain name the traffic levels in the responses have increased by a factor of 5.5. When the DNSSEC-signatures cannot be validated the client will repeat the query on any alternate DNS resolvers that have been configured. One way to look at this is to compare it to the validly signed domain. DNSSEC-invalidity is observed to increase the total response traffic volume by 20%. But this condition is being encountered by at most 4% of clients. If every client was using resolvers that performed DNSSEC validation then the consequence of key expiration, or any other event that caused the signature information be become invalid, would increase the traffic levels by 500%. In other words, the total traffic volume would be 6 times greater than that of a validly signed domain, or some 96 times higher than that of a validly signed domain, when using a single name server in the case where none of the responses are cached in DNS resolvers.

Conclusions

If you DNSSEC-sign your domain, then it would be prudent to assume that the small fraction of client's DNS resolvers that currently perform DNSSEC signature validation will only increase over time. If you want to plan for a time when the entirety of the DNS resolution function includes the capability to perform DNSSEC validation, then signing your domain should entail a prudent provision for ensure that your authoritative name servers can handle 4 times the query volume for an unsigned version of the domain, and some 15 times the response traffic volumes.

But while it's good to plan for success, you should perhaps also factor in a plan for failure. Even if you operate with precision and care in your own DNSSEC key management and signing practices in zone administration, the validity of DNSSEC validation operations depends on an equal level of precision, care, and above all stability, all the way to the DNS root keys. If the DNS root keys become invalid for any reason, or if the inter-linked chain of DS and DNSKEY key values is broken in any of your ancestor DNS zones, then your domain becomes invalid in DNSSEC terms, despite your own efforts on local zone management, and you should plan for this eventuality. The consequences of failure are serious. In our experiment we've used a single name server, and the figure point to a conclusion that if you want to plan for a time when the entirety of the DNS resolution function includes the capability to perform DNSSEC validation, then making provision for signing your domain should entail a prudent provision for ensure that your authoritative name servers can handle 10 times the query volume that would be experienced for an unsigned version of the domain, and a total response traffic volume of 100 times greater. If you have multiple name servers for your domain, and many domains do, then the traffic volumes for the domain when the DNSSEC signature is invalid would be expected to rise proportionately. This is probably a fruitful area for further measurement in DNSSEC behaviour modes.

For clients, the penalty is one of adding the time to undertake additional queries for DNSKEY and DS RRs. Given the serial nature of the query behaviour this can add some delay in name resolution, but this would be conventionally anticipated to be offset by cache behaviour, where the DNSSEC RRs are per zone rather than terminal names, so DNS caching is potentially highly efficient for DNSSEC.

However, failure is the issue here. When a client's DNS resolver encounters a failed DNSSEC signature validation chain, then the DNS resolver will requery the other name servers, and the client will requery the alternate DNS resolvers. This all takes additional time, and in the case of failure we observe that failure takes not just additional milliseconds, but for a large proportion of the client base some additional seconds to complete the name resolution process.

The issue here is that in DNSSEC the consequences of failure are extremely severe. Severe both in terms of causing client DNS resolvers to thrash, and severe in terms of dramatic increases in the load imposed on authoritative name servers. It has probably been said before in other contexts, but in the case of DNSSEC what we would really appreciate right now is a better form of failure!

Disclaimer

Authors

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.
www.potaroo.net

George Michaelson B.Sc., is a research scientist at the Asia Pacific Network Information Centre (APNIC), the Regional Internet Registry serving the Asia Pacific region. George explores problems in Internet Number Resource management, Internet standards, and network measurement by collaborative research. George has more than 30 years experience in computer science, networking, ICT administration, and research conducted in Australia and the UK. He participates in standards development in the IETF and has been a working group chair as well as an RFC author. He is a member of the British Computer Society.