April 2013
Geoff Huston

DNSSEC and Google's Public DNS Service

The Domain Name System, or the DNS, is a critical, yet somewhat invisible component of the Internet. The world of the Internet is a world of symbols and words. We invoke applications to interact with services such as Google, Facebook and Twitter, and the interaction is phrased in human readable symbols. But the interaction with the network is one that is entirely in a binary format. So our symbolic view of a service, such as **`www.google.com`**, has to be translated into a protocol address, such as 74.125.237.144. The mapping from symbols to protocol addresses is one of the critical functions of the DNS. We rely not only on the continued presence of the DNS, but its correct operation as well. Entering **`mybank.com.au`** in a browser does not necessarily guarantee that your interaction will be with your intended service. One of the more insidious attack vectors for the Internet is to deliberately corrupt the operation of the DNS, and thereby dupe the user's application to open a session with the wrong destination. The most robust response we've managed to devise to mitigate this longstanding vulnerability in the DNS has been to add secure cryptographic signatures into the DNS, using a technology called DNSSEC.

The story of DNSSEC has strong similarities to that of IPv6. Like IPv6, DNSSEC has been around for many years, but its languishing. Like IPv6, DNSSEC is most effective when everyone is using it, and the marginal returns from piecemeal adoption are extremely low. And like IPv6, the relatively low levels of deployment and use of DNSSEC does not reflect the longstanding effort to lift the visibility of the technology and concerted efforts to publicise the clear long term benefits in the use of this technology.

Even when the insidious dangers of an attack on the integrity of the DNS and the widely used Domain Name Certificate infrastructure were clearly demonstrated in the Diginotar incident in 2011 (http://www.potaroo.net/ispcol/2011-10/hacking.html), the potential benefits of the use of DNSSEC and the adoption of a secure binding of the domain name to IP address through digital credentials placed in a secure DNS framework failed to gain traction in the Internet. In some ways there's a form of mutual deadlock going on: While so few clients use DNSSEC, there appears to be little motivation on the part of domain name admins to use DNSSEC to sign domain names. And while there are so few signed domain names, there is little motivation to deploy client tools to incorporate DNSSEC validation into DNS resolution. And while domain names are largely unsigned and so few clients use DNSSEC to validate DNS name resolution outcomes, there is no motivation for domain name admins or browser authors to use the DANE technology to provide a secure form of mapping from a domain name to IP address and a public TLS key.

Can we quantify the extent to which DNSSEC has been deployed and used in today's Internet? One way is to measure the extent to which domain names are signed using DNSSEC signatures. Services such as SecSpider (http://secspider.cs.ucla.edu/growth.html) track the progress of domain signing across the various top level and second level DNS zones by counting the number of domain names that include DNSSEC credentials. But the other half of the question is also relevant here. To what extent do end user applications use DNS resolvers that perform DNSSEC validation when resolving a name? And, most critically, to what extent will end user applications refrain from using a DNS name resolution result if the domain name is DNSSEC-signed, and the associated DNSSEC signature validation fails?

At APNIC Labs we've been looking at this second question, attempting to quantify the extent to which clients use DNSSEC validation in conjunction with name resolution. Our initial efforts were undertaken in October and November 2012. The first measurement exercise in October 2012 pointed to some 9% of clients who appeared to use DNS resolvers that were seen to perform some form of DNSSEC validation. A re-run of the experiment in November 2012 provided the result that some 1.6% of clients who appeared to exclusively use DNS resolvers that consistently performed DNSSEC validation (http://www.potaroo.net/ispcol/2012-10/counting-dnssec.html, http://www.potaroo.net/ispcol/2012-10/counting-dnssec-2.html).

Neither of these results were all that satisfying, and the cause of this level of disquiet with the results was attempting to factor in the effects of DNS resolver caching into the experiment's results. In both these experiments we used an online ad campaign to enrol millions of end clients to perform a collection of objects to retrieve, and in both cases we used wildcards in the DNS and the dynamic generation of unique DNS labels to present each client with a set of unique DNS names to resolve. However, the exposed a weakness in this approach, in so far as a single DNSSEC signature chain was shared across the experiment. This meant that once a DNS resolver had retrieved the DNSSEC Resource Records (RRs) for the experiment's DNS names (the DNSKEY and DS RRs of the signed zone containing the wildcard entry) it served all subsequent DNSSEC RR queries from its local cache, only refreshing the cached data upon expiry of the cache lifetime of the stored records (which was set to one hour). Given that our observation point for the experiment was the DNS query log at the authoritative name server, we were forced to infer the DNSSEC capabilities of each of the visible DNS resolvers based on the sequence of DNS queries as seen at the authoritative name server and the timings between queries. These experiments found an approximate upper bound of around 9% of clients using DNSSEC validating resolvers using a rather liberal test for DNSSEC validation, and an approximate lower bound of 1.6% of clients who used DNSSEC validating resolvers using a much stricter set of constraints of inference rules. Obviously we are interested in reducing the uncertainty in these measurements if we can.

In reviewing the previous experiments we noted that one way to remove the need to infer DNS resolver behaviour was to use a DNS configuration that removed, as much as possible, the effects of DNS resolver cache behaviour. Rather than use a simple wildcard in a common DNSSEC-signed domain, if we could present to each client a unique DNSSEC-signed domain, then the unique label would force any DNSSEC-validating resolver to retrieve the DNSSEC RRs from the authoritative name server for this unique DNS zone. In other words, if the client's resolvers were performing DNSSEC validation then the authoritative name server would not only receive the A queries for the address of the domain name, but also receive the DNSKEY and DS queries as part of the DNSSEC validation phase. This approach would allow us to ascertain with a greater level of accuracy how many clients were using DNSSEC to validate DNS names. This DNS name structure is shown in Figure 1.

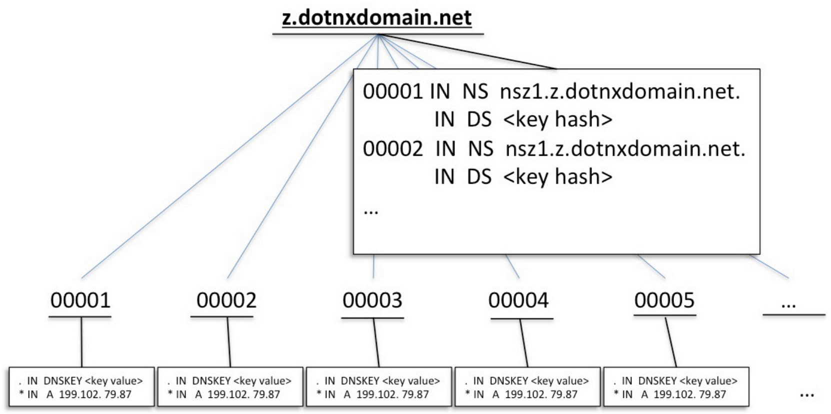

Figure 1: Domain Name Structure used to Measure DNSSEC Validation behaviour

We ran this version of the DNSSEC measurement experiment from the 8th to the 18th March 2013.

On the 19th March Google announced (http://googleonlinesecurity.blogspot.com.au/2013/03/google-public-dns-now-supports-dnssec.html) that their Google Public DNS resolvers supported DNSSEC validation. In their announcement, Google reported that:

> "Currently Google Public DNS is serving more than 130 billion DNS queries on average (peaking at 150 billion) from more than 70 million unique IP addresses each day. However, only 7% of queries from the client side are DNSSEC-enabled (about 3% requesting validation and 4% requesting DNSSEC data but no validation) and about 1% of DNS responses from the name server side are signed."

This announcement appeared to present an ideal opportunity for a "before and after" exercise, in performing the DNSSEC measurement exercise in the period immediately following Google's apparent switch to its resolvers to perform DNSSEC validation. Just what impact would this switch have on the overall picture of clients performing DNSSEC validation? We re-ran the same experiment from the 22nd March to the 1st April 2013 in order to measure the extent that this announcement of a shift by Google's Public DNS servers has changed the overall picture of clients who use DNSSEC validation.

At this time Google has not turned on DNSSEC validation unconditionally. To quote from additional material published by Google (https://developers.google.com/speed/public-dns/faq#dnssec):

> **Does Google Public DNS support the DNSSEC protocol?**
>
> Yes. Google Public DNS is a validating, security-aware resolver. Currently this is an opt-in feature: for queries coming from clients requesting validation (the AD and/or DO flag is set), Google Public DNS verifies that response records are correctly authenticated. Validation by default (i.e. for all queries) will be enabled soon.
>
> **Which client resolvers currently enable DNSSEC?**
>
> Unfortunately, most standard client stub resolvers do not enable full DNSSEC checking and cannot be easily reconfigured to do so. We have decided to make our initial launch only cover resolvers that explicitly ask for DNSSEC checking so that we become aware of any problems before exposing our users to possible large-scale DNS failures due to DNSSEC misconfigurations or outages. Once we are happy that we can safely enable DNSSEC for all users except those who explicitly opt out, we will do so.

The implication is that, at present, if the client DNS query does not request DNSSEC validation, then the Google Public DNS will return a result without performing any form of DNSSEC validation of the response. The Google material indicates that a DNSSEC validation request is marked by a DNS query having the AD or the DO flag set.

What are these DNS query flags?

DNSSEC and DNS Query Flags

The DNS protocol, as defined in RFC1034 and RFC 1035, did not include any specific provision for validation of DNS responses. The protocol is a simple query/response protocol. The client generates a DNS query consisting of a completed header and a query section, and the response contains the same header and query sections together with answer, authority and additional sections.

DNSSEC is a backward compatible extension to the DNS protocol, defined in RFCs 4033, 4034, 4035, 5155 and 6840. There are three flags in a DNS query that contain explicit DNSSEC instructions to a resolver.

The DO flag

The first is part of the Extended Mechanisms for DNS (EDNS0), defined in RFC 2671. If the DNSSEC OK (DO) bit is set in a query, then this is interpreted as a signal that the response should include DNSSEC data in its response. If the response is a Resource Record from a DNSSEC-signed zone, then the response should include the signature of the Resource Record (RRSIG RR) in additional to the Resource Record response. If there is no such domain, then the response should include NSEC or NSEC3 RRs in its response.

The CD flag

The second is the Checking Disabled (CD) flag. If this flag is set in a query then the resolver should return the requested information, including the RRSIG records as appropriate if the DO flag is set, but should not attempt to validate the signatures included in the response. By setting the CD bit in its query the originating resolver is indicating that it is taking responsibility for performing authentication of the response, and that the recursive name server being queried should not interfere with this function.

The AD flag

The last flag is the Authenticated Data (AD) flag. This flag has a somewhat unclear meaning. Up until the start of 2013 the standards document was RFC4035:

```
4.6.  Handling of the CD and AD Bits
   …
   A security-aware resolver MUST clear the AD bit when composing query
   messages to protect against buggy name servers that blindly copy
   header bits that they do not understand from the query message to the
   response message.
```

In February RFC6840 was published, with the following re-definition of the AD bit:

```
5.7.  Setting the AD Bit on Queries

   The semantics of the Authentic Data (AD) bit in the query were
   previously undefined.  Section 4.6 of [RFC4035] instructed resolvers
   to always clear the AD bit when composing queries.

   This document defines setting the AD bit in a query as a signal
   indicating that the requester understands and is interested in the
   value of the AD bit in the response.  This allows a requester to
   indicate that it understands the AD bit without also requesting
   DNSSEC data via the DO bit.

5.8.  Setting the AD Bit on Replies

   Section 3.2.3 of [RFC4035] describes under which conditions a
   validating resolver should set or clear the AD bit in a response.  In
   order to interoperate with legacy stub resolvers and middleboxes that
   neither understand nor ignore the AD bit, validating resolvers SHOULD
   only set the AD bit when a response both meets the conditions listed
   in Section 3.2.3 of [RFC4035], and the request contained either a set
   DO bit or a set AD bit.
```

Combined Flag Settings

The effect of various combinations flag settings in queries sent to resolvers is shown in the following table.

DO	CD	AD	Effect
0	0	0	The resolver may or may not perform DNSSEC validation. No DNSSEC RRs are passed back in the response.
0	0	1	The resolver should perform validation. No DNSSEC RRs are passed back in the response
0	1	0	The resolver should not perform DNSSEC validation. No DNSSEC RRs are passed back in the response
0	1	1	Mixed signals! The resolver's actions are undefined.
1	0	0	The resolver should perform validation, and return DNSSEC RRs in its response
1	0	1	The resolver should perform validation, and return DNSSEC RRs in its response
1	1	0	The resolver should not perform DNSSEC validation, but it should return DNSSEC RRs in its response
1	1	1	Mixed signals! The resolver's actions are undefined.

Table 1: DNSSEC Flag settings in DNS queries

It should be noted that when the resolver is directed not to perform DNSSEC validation, then the resolver should respond with the requested resource records, even in the case that the resource records are DNSSEC-signed and the signature is invalid.

DNS Resolver Behaviour and DNSSEC

The simplest conceptual model of DNS resolution involving a client, a DNS resolver and a collection of authoritative name servers, as shown in Figure 2.

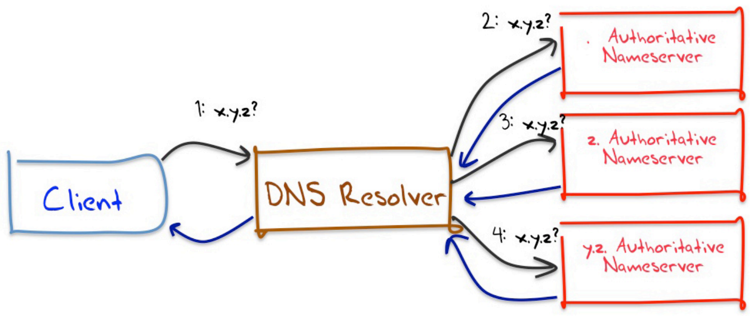
2: x.y.z?
. Authoritative Nameserver
1: x.y.z?
3: x.y.z?
z. Authoritative Nameserver
Client
DNS Resolver
4: x.y.z?
y.z. Authoritative Nameserver

Figure 2 – Simple Model of DNS Resolution for the domain name "x.y.z"

When the client queries its DNS resolver for resolution of a domain name, such as **x.y.z**, then the DNS resolver, assuming that it has a cleared cache state, would first send this query to a root name server. The server would respond with the set of authoritative name servers for the top level domain **z.**. The DNS resolver would then send the same query to one of these name servers for **z.**, which would respond with the name servers for the domain **y.z.**. The DNS resolver will then query one of these name servers for the DNS name **x.y.z.**, and it will then pass the response it receives back to the client. The sequence of DNS queries sent by this DNS resolver after receiving the initial query from the client would be as follows:

#	Query RR	Name Server	Response
1.	A x.y.z.	.	NS for "z."
2.	A x.y.z.	z.	NS for "y.z."
3.	A x.y.z.	y.z.	A for x.y.z.

Figure 3 – DNS queries without DNSSEC validation

What if this DNS resolver was a DNSSEC validating resolver? This would involve additional DNS queries, as the DNS resolver needs to validate the RRSIG record for the retrieved resource record. This validation involves a "back trace" up the DNS delegation hierarchy, following the chain of **DNSKEY** and **DS** RRs until it reaches a trust point, which, in this case, is the signing key for the root zone. The equivalent set of DNS queries for a DNSSEC validating resolver, after receiving the initial query from the client, would be as follows:

#	Query RR	Name Server	Response
1.	A EDC x.y.z.	.	NS for "z." + RRSIG
2.	A EDC x.y.z.	z.	NS for "y." + RRSIG
3.	A EDC x.y.z.	y.z.	A for x.y.z. + RRSIG
4.	DNSKEY EDC y.z.	y.x	DNSKEY for zone "y.z" + RRSIG
5.	DS EDC y.z.	z.	DS for zone "y.z" + RRSIG
6.	DNSKEY ECD z.	z.	DNSKEY for zone "z." + RRSIG
7.	DS EDC z.	.	DS for zone "z." + RRSIG
8.	DNSKEY EDC .	.	DNSKEY for zone "."

Figure 4 – DNS queries with DNSSEC validation

The initial 3 queries are similar, but now the queries emitted by this DNSSEC-validating DNS resolver include the EDNS0 (**E**) extension, and the DNSSEC OK (**D**) and the Checking Disabled (**C**) flags. In each case the **DO** flag signals that the responses from the authoritative servers are to include the **RRSIG** RRs, if the zones are indeed DNSSEC-signed.

If we were to look at the transactions on the authoritative server for zone **y.z.** we would see an **A** query followed by a **DNSKEY** query. If we were look at the transactions on the authoritative server for the parent zone **z.**, there would be an **A** query, followed by a **DS** query and a **DNSKEY** query. And if we were to use the same authoritative name server for both **z.** and **x.z.** then this name server would see an **A** query for **x.y.z.**, followed by **DNSKEY** and **DS** queries for **y.z.**

From this example it would appear feasible to categorize DNS resolvers into *DNSSEC-validating* and *non-validating* resolvers based on the observation of DNSKEY and DS queries at the authoritative name server. If we observe queries as in Figure 3 then the DNS resolver is a *non-validating* resolver, and if they are as in Figure 4, then the resolver is a *DNSSEC-validating* resolver. Unfortunately this simple form a categorization is not possible, as the picture of DNS resolution can be far more complex that the simple picture of Figure 2.

A DNS resolver can be configured to use a "forwarder". In this case the DNS resolver will not query the authoritative name servers directly, but channel its queries through another DNS resolver. The advantage of using a forwarder is to leverage the efficiency of caching of DNS results. Variants of this resolver configuration allow the forwarding function to be limited to the resolution of specific DNS zones rather than apply it to all queries. A resolver can also be configured to fall back to recursive resolution if the forwarders fail to provide an answer. There are also "resolver farms" in the DNS, where a single logical forwarder may accept DNS queries, but then use a collection of "slave" DNS resolvers to undertake the queries. The idealized picture of DNS resolution in Figure 2 should be augmented somewhat to illustrate the level of potential complexity involved with DNS resolution. Figure 5 shows some of the possible configurations.

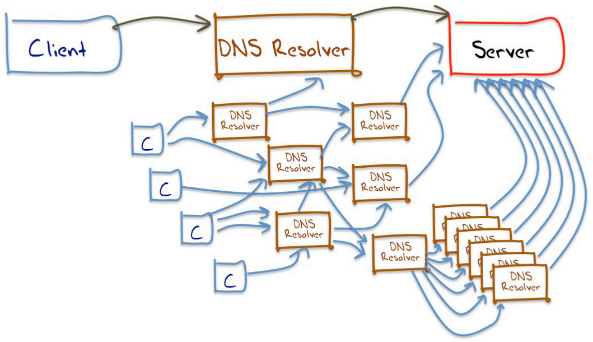

Figure 5 – Some DNS Resolution structures with Forwarders

In the context of the experiment we are undertaking here, we have instrumented only the authoritative name server, and we only see the DNS queries submitted by "visible" DNS resolvers to this server.

Thus, in terms of directly visible resolvers, the view of the DNS resolution infrastructure is closer to that as shown in Figure 6. Here the internal structure of DNS forwarders is effectively occluded from the authoritative name server, and the server can only see those resolvers that pass queries to it.

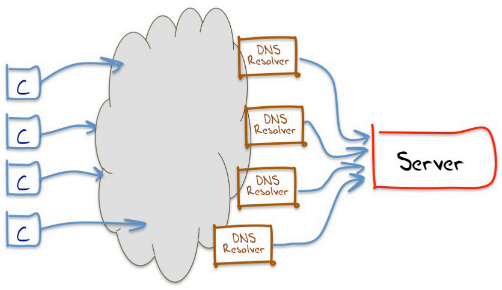

Figure 6 –DNS Resolution structures

The implication here is that while this experimental approach can provide a good view of the DNSSEC validation capabilities of clients, it requires some level of inference, and an associated level of uncertainty of the application inference rules, to infer the DNSSEC validation behavior of DNS resolvers. For example, if a DNSEC-validating resolver uses a non-validating forwarder, then the authoritative name server will see a sequence of queries from the forwarder with the DO and CD flags set that are consistent with DNSSEC validation. If a non-DNSSEC-validating resolver uses as a forwarder a DNSSEC-validating resolver, then the authoritative name server may see precisely the same sequence of queries from this forwarder.

The DNSSEC Measurement Experiment

In this experiment an Adobe Flash object is embedded in an online ad. The code is executed upon presentation of the advertisement, and does not require the user to click on the ad impression.

The code causes the client to retrieve an object from an experiment controller, which feeds the client with 4 URLs to fetch. The first three URLs are to be fetched immediately, while the fourth URL is to be fetched once the first three objects have been fetched, or upon the expiration of a 10 second timer, whichever occurs first. All of these URLs include a unique DNS label. An example set of the four URLs is shown below. The URL generator generates a set of unique values for each experiment. These unique fields are highlighted in color.

```
d
      http://d.u7280280162.s1364784185.v6022.69da1.z.dotnxdomain.net/1x1.png?d.u72802
      80162.s1364784185.v6022.69da1.z.dotnxdomain.net

e
      http://e.u7280280162.s1364784185.v6022.69da1.z.dashnxdomain.net/1x1.png?e.u7280
      280162.s1364784185.v6022.69da1.z.dashnxdomain.net

f
      http://f.u7280280162.s1364784185.v6022.69da2.z.dotnxdomain.net/1x1.png?f.u72802
      80162.s1364784185.v6022.69da2.z.dotnxdomain.net

results
      http://results.u7280280162.s1364784185.v6022.69da1.x.rand.apnic.net/1x1.png?res
      ults.u7280280162.s1364784185.i767.v6022.69da1&r=
```

As shown in the above example, three URLs share a hexadecimal DNS label, "`69da1`" in this instance, while the other URL includes a label that is one greater in value, "`69da2`" in this instance. The first URL, starting with the **d** label, uses a combination of a wildcard record and DNSSEC-signed domain. In this case the DNSSEC signature of the A resource record is valid. The second URL, starting with the **e** label, uses the same combination, but in this case the domain is not DNSSEC-signed. The third URL, starting with the **f** label, also uses a combination of a wildcard domain and DNSSEC-signed domain. However, in this case the DNSSEC signature is invalid, in so far as the DS record for the zone `69da2.z.dotnxdomain.net` does not match the corresponding DNSKEY records. The **results** URL is not DNSSEC signed. The order of the presentation of the URLs in the script is fixed as the sequence <**d, e, f, results**> with the final URL delayed for either 10 seconds, or the successful fetching of the first three URLs, whichever occurs first. The client is instructed to start timers for the fetching of the **d**, **e** and **f** URLs, and report the timer values in the **results** URL.

Our intention was to ensure that each client was presented with a uniquely signed DNS label, ensuring that there was no cached state in any DNS resolver or in any Web Proxy server. The result of this is that each experiment causes a set of DNS and HTTP transactions with the authoritative DNS servers and the web servers for these domain names.

The experiment uses a single server, which contains both a DNS name server (running BIND 9.9.2-P1), and a web server (running Apache V2.2.23). This server is the only authoritative name server for the domains used in these four URLs, and the A RRs point only to this server. In this experiment we deliberately set up the entire experiment using IPv4, leaving investigation of IPv6 and DNSSEC to other experiments.

While the DNSSEC validation capabilities of individual DNS resolvers are somewhat challenging to infer from this form of experiment, it we take a step back and pose the question of whether the end user uses DNSSEC validation via its configured DNS resolvers, then this is possible to infer from the log of DNS queries and HTTP fetches at the common server. An example of an extract from the DNS and HTTP logs from the server, relating to a single experiment (`u5158122700.s1364428847` in this case) is shown below.

Server	**Query**
DNS	-ED IN **A e**.u5158122700.s1364428847.v6022.5e3bf.z.dashnxdomain.net
DNS	-ED IN **A f**.u5158122700.s1364428847.v6022.5e3c0.z.dotnxdomain.net
DNS	-ED IN **A d**.u5158122700.s1364428847.v6022.5e3bf.z.dotnxdomain.net
HTTP	GET/crossdomain.xml **e**.u5158122700.s1364428847.v6022.5e3bf.z.dashnxdomain.net
HTTP	GET /1x1.png **e**.u5158122700.s1364428847.v6022.5e3bf.z.dashnxdomain.net
HTTP	GET /crossdomain.xml **d**.u5158122700.s1364428847.v6022.5e3bf.z.dotnxdomain.net
HTTP	GET /1x1.png **d**.u5158122700.s1364428847.v6022.5e3bf.z.dotnxdomain.net
HTTP	GET /crossdomain.xml **f**.u5158122700.s1364428847.v6022.5e3c0.z.dotnxdomain.net
HTTP	GET /1x1.png **f**.u5158122700.s1364428847.v6022.5e3c0.z.dotnxdomain.net
DNS	-ED IN **A results**.u5158122700.s1364428847.v6022.5e3bf.x.rand.apnic.net
HTTP	GET /crossdomain.xml **results**.u5158122700.s1364428847.v6022.5e3bf.x.rand.apnic.net
HTTP	GET /1x1.png?**results**.u5158122700.s1364428847.i767.v6022.5e3bf&r=zd-1923.ze-1578.zf-1578

Figure 7 – Server Logs from an experiment that did not perform DNSSEC validation

This experiment generated 4 DNS queries, and 8 HTTP queries. The DNS queries all used the DNSSEC OK (DO) flag ("**-ED**" in the log indicates that recursive resolution of the query was disabled (**-**), EDNS0 was being used (**E**), and DNSSEC signature Resource Records were requested, if available (**D**)). However, the DNS resolver is asking for **A** RRs but not asking for **DNSKEY** or **DS** RRs. It appears that while the resolver has set the **DNSSEC OK** flag, it is not performing any form of DNSSEC validation on the value returned in the RRSIG RRs.

The client retrieved all three (**d**, **e** and **f**) URLs. Considering that the **f** URL has an invalid DNSSEC signature then this confirms that the DNS resolver being used by this client does not perform any DNSSEC validation. The result line also includes a client-side timer result, and the client is reporting that experiment **d** took the client 1,923ms to load, **e** took 1,578ms and **f** took 1,578ms. The slightly longer time to perform the **d** URL fetch as compared to the **f** URL fetch could be due to the resolver's caching of the resolution of **z.dotnxdomain.net** while resolving the **d** URL, and using the cached values when resolving the **f** URL.

Now, by contrast, the following is an extract from the same logs that show the DNS queries received by the authoritative name server from a DNS resolver that appears to be performing DNSSEC validation

Server	Query
DNS A	-EDC IN **A d**.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
DNS A	-EDC IN **A e**.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
DNS A	-EDC IN **DNSKEY** 5e4e3.z.dotnxdomain.net
DNS A	-EDC IN **DS** 5e4e3.z.dotnxdomain.net
DNS A	-EDC IN **A f**.u94278337.s1364428957.v6022.5e4e4.z.dotnxdomain.net
DNS A	-EDC IN **DNSKEY** 5e4e4.z.dotnxdomain.net
DNS A	-EDC IN **DS** 5e4e4.z.dotnxdomain.net
DNS B	-EDC IN **A f**.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
DNS B	-EDC IN **DNSKEY** 5e4e4.z.dotnxdomain.net
DNS B	-EDC IN **DS** 5e4e4.z.dotnxdomain.net
HTTP	GET /crossdomain.xml **d**.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
HTTP	GET /1x1.png **d**.u94278337.s1364428957.v6022.5e4e3.z.dotnxdomain.net
HTTP	GET /crossdomain.xml **e**.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
HTTP	GET /1x1.png **e**.u94278337.s1364428957.v6022.5e4e3.z.dashnxdomain.net
DNS A	-EDC IN **A results**.u94278337.s1364428957.v6022.5e4e3.x.rand.apnic.net
HTTP	GET /crossdomain.xml **results**.u94278337.s1364428957.v6022.5e4e3.x.rand.apnic.net
HTTP	GET /1x1.png?**results**.u94278337.s1364428957.i767.v6022.5e4e3&r=zd-1572.ze-1269.zf-null

Figure 8 – Server Logs from an experiment that performed DNSSEC validation

In this case, the client retrieved two URLs (**d** and **e**), and did not retrieve the URL associated with the **f** domain name that failed DNSSEC validation.

The DNS resolver is performing DNSSEC validation, as evidenced by the retrieval of the **DNSKEY** and **DS** RRs for the **d** and **f** domain zones. The first DNS resolution for the **f** domain name failed DNSSEC validation, and the client appears to have received a SERVFAIL response. The client then tried its second configured resolver, which also performs DNSSEC validation. At this stage the client gave up on attempting to resolve the **f** domain name. After the expiration of the 10 second timer the client fetched the **result** URL. The result line also includes a client-side timer result, and the client is reporting that the **d** URL took the client 1,572ms to load, **e** took 1,269ms and **f** was not retrieved. The slightly longer time to perform the **d** fetch as compared to the **e** fetch could be due in part to the resolver's retrieval of the DNSKEY and DS RRs for the **d** domain, and the associated DNSSEC validation operation that was performed by the validating resolver, as the client would not receive the result of the **d** DNS query until and the additional queries associated with DNSSEC validation had been completed.

The aim of this experiment is to sample a large random set of clients from all over the Internet and get their browser to execute this experiment. If the set of transactions at the server resembles the first set of transactions shown above then we can conclude and the client is not using DNSSEC, while if the set of transactions resembles the second set shown above, then the client is using DNS resolvers that perform DNSSEC validation to protect its name resolution function.

In this experimental setup the only accessible component that we can equip with instrumentation is the authoritative server for the domain "dotnxdomain.net.". If the DNS resolver was not performing DNSSEC validation then we would expect to see a single query for an **A** RR made to the authoritative server, while if the DNS resolver was a DNSSEC validating resolver then we would expect to see an **A** RR query, followed by a **DNSKEY** and a **DS** query.

And this is what is seen. The following queries were logged by the authoritative name server from a non-validating DNS resolver:

```
22-Mar-2013 01:46:34.209 10.0.0.1#59296: query: e.u3435434437.s1363916793.v6022.58cb7.z.dashnxdomain.net IN A -ED
22-Mar-2013 01:46:34.311 10.0.0.1#57571: query: d.u3435434437.s1363916793.v6022.58cb7.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:46:34.245 10.0.0.1#6322e: query: f.u3435434437.s1363916793.v6022.58cb8.z.dotnxdomain.net IN A -ED
```

The following queries were logged by the authoritative name server from a set of DNS resolvers that appear to perform DNSSEC validation:

```
22-Mar-2013 01:40:48.029 10.0.0.2#17283 query: d.u2867716218.s1363916447.v6022.58721.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.038 10.0.0.2#22572 query: 58721.z.dotnxdomain.net IN DS -ED
22-Mar-2013 01:40:48.056 10.0.0.2#54384 query: 58721.z.dotnxdomain.net IN DNSKEY -ED
22-Mar-2013 01:40:48.229 10.0.0.3#18024 query: e.u2867716218.s1363916447.v6022.58721.z.dashnxdomain.net IN A -ED
22-Mar-2013 01:40:48.024 10.0.0.4#19869 query: f.u2867716218.s1363916447.v6022.58722.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.032 10.0.0.4#52521 query: 58722.z.dotnxdomain.net IN DS -ED
22-Mar-2013 01:40:48.049 10.0.0.4#35302 query: 58722.z.dotnxdomain.net IN DNSKEY -ED
22-Mar-2013 01:40:48.084 10.0.0.5#44543 query: f.u2867716218.s1363916447.v6022.58722.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.092 10.0.0.5#46424 query: 58722.z.dotnxdomain.net IN DS -ED
22-Mar-2013 01:40:48.109 10.0.0.5#17802 query: 58722.z.dotnxdomain.net IN DNSKEY -ED
22-Mar-2013 01:40:48.139 10.0.0.6#24334 query: f.u2867716218.s1363916447.v6022.58722.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.147 10.0.0.6#18014 query: 58722.z.dotnxdomain.net IN DS -ED
22-Mar-2013 01:40:48.164 10.0.0.6#43916 query: 58722.z.dotnxdomain.net IN DNSKEY -ED
22-Mar-2013 01:40:48.197 10.0.0.6#51221 query: f.u2867716218.s1363916447.v6022.58722.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.230 10.0.0.7#58295 query: f.u2867716218.s1363916447.v6022.58722.z.dotnxdomain.net IN A -ED
22-Mar-2013 01:40:48.239 10.0.0.7#34658 query: 58722.z.dotnxdomain.net IN DS -ED
22-Mar-2013 01:40:48.255 10.0.0.7#31055 query: 58722.z.dotnxdomain.net IN DNSKEY -ED
```

In this case the **f** URL, which contains an invalid DNSSEC signature, apparently generates a SERVFAIL error response from the initial DNS query that, in turn, triggers the client's name resolution process to retry the entire DNS query using a different resolver. A repetition of the same failure causes the client to perform a third and final retry on another configured resolver.

Experiment Results

The first run of this experiment had the following results:

Presented Experiments:	2,802,390
Presented Experiments with Web Fetches:	2,632,322
Presented Experiments with Result Web Fetch:	2,142,141

Table 2 – Experiment A Counts

This shows the "drop off" rate in the experiment. When the ad is presented to the user (an "impression") the end user's browser is passed a Flash object. Execution of the Flash object causes the end-user's browser to collect the parameters for the experiment, which consists of the three experiment URLs and the result URL. The browser will then commence fetches of the three URLs by resolving the DNS names for the URLs. The first measurement of "Presented Experiments" at the authoritative name server is the total count of the number of unique identifiers that generated DNS queries. The end user browser will then perform fetches of the URLs. The difference between the DNS query count and the Web fetch count shows that some 6% of experiment runs are aborted before proceeding to the Web fetch part of the experiment. When either all three URLs have been fetched, or 10 seconds have elapsed, the end user's browser will then perform a DNS resolution query for the result URL and then fetch the URL. A further 19% of experiment runs do not get to this result fetch phase. In order to ensure that the effect of aborting the experiment run is minimized, the following summary relates only to those 2,142,141 experiments that have completed the fetch of the result URL.

The experiment consists of fetching the d, e and f URLs. The following table shows the number of clients who fetched various combinations of these URLs (and also fetched the **result** URL).

d	e	f	Count
No	No	No	4,325 (0.20%)
Yes	No	No	1,024 (0.05%)
No	Yes	No	2,627 (0.12%)
Yes	**Yes**	**No**	**60,988 (2.85%)**
No	No	Yes	1,045 (0.05%)
Yes	No	Yes	6,108 (0.29%)
No	Yes	Yes	3,291 (0.15%)
Yes	**Yes**	**Yes**	**2,059,990 (96.17%)**

Table 3 – Experiment A URL fetch combinations

There were a total of 60,998 experiments that fetched **d** and **e**, but not **f**. It is possible that this is due to the client being unable to validate the DNSSEC signatures on the f domain name, but it is also a possibility that the client simply did not fetch **f** due to some form of premature end of the experiment's execution, even though the result URL had been fetched. It is possible to use the DNS logs to confirm whether or not the client attempted to validate the **f** domain name by looking for those clients who retrieved the DS and DNSKEY RRs for both **d** and **f**, but only fetched **d** and **e.** As shown in the table below, there were 58,303 such experiment runs, or 2.72% of the total.

We are also interested in the number of clients who use multiple DNS resolvers where only some of the resolvers perform DNSSEC validation. In the event that a DNSSEC-validating resolver returns the SERVFAIL error code it is likely that a client will then pass the query to other resolvers. If these resolvers do not perform DNSSEC validation then the client will receive a response for the **f** domain name. So the next category is the number of clients who fetched **d**, **e** and **f**, and fetched the DS and DNSKEY RRs for both **d** and **f**. These users appear to be using a mix of DNSSEC-validating and non-DNSSEC validating clients. As shown in the table below, there were 52,713 such experiment runs, or 2.46% of the total.

Of the remaining experiments we have extracted out those experiment runs where only A RRs were queried, which accounted for 2,026,014 experiment runs, or 94,58% of the total.

What is left are some 2,368 experiment runs which appears to fetch some DNSSEC RRs, but could not be classified into either of the above two categories.

Used DNSSEC Validating Resolvers:	58,303	2.72%
Used a mix of validating and non-validating resolvers:	52,713	2.46%
Fetched DNSSEC RRs some of the time:	2,368	0.11%
Did not fetch any DNSSEC RRs:	2,026,014	94.58%

Table 4 – Experiment A Client Capabilities for DNSSEC

The result of this experiment shows that some **2.72%** of all clients appear to exclusively use DNSSEC validating resolvers, and will be unable to resolve a DNS name when the DNSSEC signature is invalid. A further **2.46%** of clients use a mix of DNSSEC-validating and non-validating resolvers, so that they do not derive any tangible "protection" from this mixed configuration.

What changed following the announcement of DNSSEC validation from Google's Public DNS resolvers? The following is the same data from the experiment that was conducted form the 22nd March to the 1st April.

Presented Experiments:	2,590,330
Presented Experiments with Web Fetches:	2,421,138
Presented Experiments with Result Web Fetch:	1,930,180

Table 5 – Experiment B Counts

d	e	f	Count
No	No	No	3,471 (0.18%)
Yes	No	No	797 (0.04%)
No	Yes	No	2,698 (0.14%)
Yes	**Yes**	**No**	**67,487 (3.50%)**
No	No	Yes	832 (0.04%)
Yes	No	Yes	4,027 (0.21%)
No	Yes	Yes	3,352 (0.17%)
Yes	**Yes**	**Yes**	**1,844,790 (95.58%)**

Table 6 – Experiment A URL fetch combinations

Used DNSSEC Validating Resolvers:	64,690	3.35%
Used a mix of validating and non-validating resolvers:	43,657	2.26%
Fetched DNSSEC RRs some of the time:	2,652	0.11%
Did not fetch any DNSSEC RRs:	1,816,968	94.13%

Table 7 – Experiment B Client Capabilities for DNSSEC

The result of this second run of the experiment shows that some **3.35%** of all clients appear to exclusively use DNSSEC validating resolvers, and, correctly, will be unable to resolve a DNS name when its DNSSEC signature is invalid. A further **2.26%** of clients use a mix of DNSSEC-validating and non-validating resolvers. The level of DNSSEC-validating clients appears to have risen by slightly more than 0.5%.

Is this positive change of the levels of clients who perform DNSSEC validation attributable to a change in the behavior of Google's Public DNS servers?

Google's Public DNS Servers

This increase of 0.5% in the number of clients performing DNSSEC validation from Experiment A to Experiment B could possibly be attributed to Google turning on DNSSC validation in its Public DNS servers at the time of the announcement, or it could be within the bounds of experimental error, or it could be the outcome of some other set of resolvers turning on DNSSEC validation in this period. It's a reasonable question to ask whether the data sets of DNS queries from Google's DNS name servers before and after Google's announcement show any difference in the extent to which they perform DNSSEC validation.

Google's Public DNS is a DNS forwarder, so in looking at the profile of DNS queries that are generated by Google resolvers, it may be useful to look at the set of transforms between queries sent to Google and queries that the Google DNS resolvers will make to the authoritative name servers. (assuming a set of unique domain names that are not already loaded into Google's local DNS cache).

Original Query	**Google's corresponding query**	
	Non-Validating	**Validating**
A	**A**	**A**
A(cd)	**A**	**A**
A(do)	**A**(do)	**A**(do), **DS**(do), **DNSKEY**(do)
A(do, cd)	**A**(do)	**A**(do)
DS(cd)	**DS**(do)	**DS**(do)
DNSKEY(do, cd)	**DNSKEY**(do)	**DNSKEY**(do)

In this table 'cd' is the Checking Disabled flag set, and 'do' is the EDNS0 option present and the DNSSEC OK flag set

Table 8 – DNS Query Transforms as performed by Google's Public DNS Forwarders

Google's Public DNS resolvers do not set the CD flag on any of the queries they make to authoritative name servers (contrary to the explicit advice in section 5.9 of RFC6840, but as the server is an authoritative name server, the advice in this RFC in this particular case is of dubious value! Google's DNS resolvers do not appear to alter any DNS resolution behaviour in not setting the CD flag when making queries to authoritative name servers).

Aside from this change to the query profile, the only other change we would expect to see from the perspective of the queries received at an authoritative name server between a Google resolver that did not perform DNSSEC validation and one that did perform validation is that the proportion of queries for an A RR with the DNSSEC OK flag set should be slightly lower, while the proportion of queries for the sequence of A, DS and DNSKEY RRs, all with the DNSSEC Ok flag should rise. The reasoning is that if Google receive a query with the DO flag set and the CD flag clear, then it will itself perform DNSSEC validation on the result, and in so doing will generate the A, DS and DNSKEY queries. If Google receive a query with the DO and CD flags set, then it will send the query to the authoritative name server with just the DO flag set, but will not perform any DNSSEC validation in its own right, as the CD flag has directed it not to do so.
So here are the results from the two runs of this measurement experiment. Experiment A is the period from the 8th to the 18th February 2013, and Experiment B is the period from the 22nd March to 1st April 2013.

	Experiment A		Experiment B	
Total number of queries passed to Google:	309,043		328,059	
Single A query:	256,708	83.07%	289,428	88.22%
Single A query with DO set:	23,134	7.48%	20,119	6.13%
Multiple A queries:	11,700	3.79%	7,916	2.41%
Multiple A queries, all with DO set:	1,179	0.38%	578	0.18%
Multiple A queries, some with DO set:	1,194	0.39%	877	0.27%
Single DNSSEC-Validate query sequence:	5,904	1.91%	3,196	0.97%
Other query sequences:	7,482	2.42%	4,706	1.43%

Table 9 – Google's Public DNS resolvers DNS Query profile

This outcome is not very illuminating. The proportion of a single DNSSEC-validation query sequence (an A query followed by DS and DNSKEY queries in either order) actually fell in Experiment B. This leads to the tentative conclusion that, within the bounds of data variation in this experiment, it appears that Google's Public DNS Servers have been performing DNSSEC validation for some time, and DNSSEC validation was not simply switched on at the time of Google's announcement on the 19th March 2013.

There is another potential way to detect Google's DNS resolver performing DNSSEC validation. If you query for an A RR from Google's public DNS servers with the DNSSEC OK bit set, and the domain name is DNSSEC signed, and the DNSSEC signature is valid, then while the first query will cause the DNS resolvers to query the authoritative name servers, subsequent queries for the same name will be answered from the DNS resolvers' cache, and will not cause any queries to be sent to the authoritative name servers. On the other hand if the DNSSEC signature of the domain name is not valid, then subsequent queries for the same domain name (with the DNSSEC OK bit set) will cause the DNS resolvers to re-query the authoritative name servers. This experiment has DNSSEC-valid and DNSSEC-invalid signatures with the **d** and **f** domain names. Are the visible differences in the query patterns from Google's Public DNS resolvers?

Category	**Experiment A**					**Experiment B**			
	Valid	**(%)**	**Invalid**	**(%)**		**Valid**	**(%)**	**Invalid**	**(%)**
Total number of queries:	153,540		154,907			164,043		165,623	
Single A query:	127,366	82.95%	128,134	82.72%		144,561	88.12%	145,725	87.99%
Single A query with DO set:	5,465	3.56%	5,386	3.48%		6,308	3.85%	6,259	3.78%
Multiple A queries:	5,557	3.62%	5,789	3.74%		3,780	2.30%	4,038	2.44%
Multiple A queries, all with DO set:	245	0.16%	248	0.16%		226	0.14%	236	0.14%
Multiple A queries, some with DO set:	202	0.13%	222	0.14%		231	0.14%	224	0.14%
Single DNSSEC-Validate query sequence:	10,818	7.05%	5,904	3.81%		6,434	3.92%	3,196	1.93%
DNSSEC Validate plus additional queries	1,864	1.21%	7,482	4.83%		1,124	0.69%	4,706	2.84%
Other	2,023	1.32%	1,742	1.12%		1,379	0.84%	1,239	0.74%

Table 10 – Google's Public DNS resolvers DNS Query profile

This table does not show any appreciable change in the query volume between the valid and invalid DNS names. His is probably due to the nature of this experiment, where the use of unique DNS labels in each experiment was deliberately designed to circumvent caching.

There is, however, one case where there is some evidence of repeated DNS queries. The drop in the relative proportion of single DNSSEC validation query sequences between the valid and invalid domain names is illustrative of a form of client behaviour where an error response of SERVFAIL from the initial DNS query causes the client to retry the same query on the next resolver in its local list of DNS name resolvers. The perhaps ironic part of this particular behaviour is that what we are seeing here in this table is the outcome of those cases where a number of DNS resolvers in the client's list of available resolvers end up forwarding these supposedly distinct queries onto the common point of Google's Public DNS servers.

So we can say with some certainty that Google did not switch on DNSSEC validation at the time of its public announcement on the 19th March 2013, and Google had been performing DNSSEC validation for some time before the announcement. It would appear that the 0.5% change in the level of end client use of DNSSEC validation is more likely the outcome of variability in the experi-ment.

The Results – Use of DNSSEC by End Users

This experiment shows that at the start of 2013 some **3%** of all clients appear to exclusively use DNSSEC validating resolvers, and, appropriately, will be unable to resolve a DNS name when its DNSSEC signature is invalid. A further **2%** of clients use a mix of DNSSEC-validating and non-validating resolvers.

Disclaimer

Author

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