May 2015
Geoff Huston

Tech Note: Measuring DNS Behaviour

The DNS is a very simple protocol. The protocol is a simple query / response interaction where the client passes a DNS transaction to a server with the query part of the transaction completed. The server fills in the answer part and possibly adds further information in the additional information part, and returns the transaction back to the client.

All very simple. What could possibly go wrong?

In practice the operation of the DNS is significantly more involved. Indeed, it's so involved that its difficult to tell whether the operation of the DNS is merely "complicated" or whether the system's interactions verge on what we would describe as "complex". When we have some simple questions about aspects of DNS behaviour sometimes they questions are extremely challenging to answer.

The question in this this instance is extremely simple to phrase. How many DNS resolvers are incapable of receiving a DNS response whose size is just inside the conventional upper limit of an unfragmented IP packet in today's Internet? Why many resolvers can successfully receive a DNS response when the combined size of the DNS response plus the protocol overheads approaches 1,500 octets in total? Before looking at how to answer this question it may be helpful to understand why this is an interesting question right now.

Rolling the Root Zone's Key Signing Key

In June 2010 the Root Zone (RZ) of the DNS was signed using the DNSSEC signing framework. This was a long-anticipated event. Signing DNS zones, and validating signed responses allows the DNS to be used in such a manner that all responses can be independently verified, adding an essential element of confidence into one of the fundamental building blocks of the Internet.

However, it goes further than that. Today's framework of security in the Internet is built upon some very shaky foundations. The concept of domain name certificates, in use since the mid-90's, was always a quick and dirty hack, and from time to time we are reminded just how fragile this system truly is when the framework is maliciously broken. We can do better, and the approach is to use DNSSEC to place cryptographic information in the DNS, signed using DNSSEC ("The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698). So these days DNSSEC is quickly becoming a critical technology. The cryptographic structure of DNSSEC mirrors the structure of the DNS name space itself, in that it is a hierarchical structure with a single common apex point, which is the anchor point of trust, namely the key used to sign across the Zone Signing Key (ZSK) that, in turn, is used to sign all entries in the root zone of the DNS. This key, the RZ Key Signing Key (KSK), is the key that was created five years ago when the RZ was first signed.

One of the common mantras we hear in cryptography is that no key should be used forever, and responsible use of this technology generally includes some regularly process to change the key value to

some new key. This was envisaged in the DNS KSK procedures, and in a May 2010 document from the Root DNSSEC Design Team ("DNSSEC Practice Statement for the Root Zone KSK Operator," http://www.root-dnssec.org/wp-content/uploads/2010/06/icann-dps-00.txt), the undertaking was made that "Each RZ KSK will be scheduled to be rolled over through a key ceremony as required, or after 5 years of operation".

Five years have elapsed, and its time to roll the RZ KSK of the DNS. The procedure is familiar: a new key pair is manufactured, and the new public key value is published. After some time the new key is used to sign the zone signing key, in addition to the current signature. Again, after some time, the old signature, and the old key is withdrawn and we are left with the new KSK in operation. A procedure to do this for the RZ KSK was documented in 2007, as RFC 5011 ("Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC5011).

There are two basic unknowns in this approach. The first is that we do not know how many resolvers implement RFC5011, and will automatically pick up the new RZ KSK when it is published in the root zone. If they do not automatically pick up the new RZ KSK via this process then they will need to do so manually. Those that do not will be left stranded when the old RZ KSK is removed, and will return SERVFAIL due to an inability to validate any signed response as their copy of the RZ KSK no longer matches the new RZ KSK. The second unknown is in the size of the DNS response, and in particular the size of the rise of the response in the intermediate stage when both KSK values are used to sign the RZ ZSK. At some stages of this anticipated key roll procedure the DNSSEC-signed response to certain DNS queries of the root, namely the RZ DNSKEY query, may approach 1500 octets in size. The first unknown is difficult to quantify in advance. However the second should measurable.

The query that is of interest here is a query for the RZ DNSKEY RR. The largest response is a scenario where the ZSK is increased to a 2048-bit RSA signature and where both the old and the new KSK values sign the ZSK. In this phase it appears that the size of the DNS response is 1,425 octets. This is compared to the current response size of 736 octets which uses a 1048-bit ZSK and a single 2048-bit KSK. This only applies to those resolvers who are requesting the DNSSEC signature data to be included in the response, which is signalled by setting the DNSSEC OK flag in the query.

How can we measure if this size of response represents an operational problem for the Internet?

DNS, UDP, Truncation, TCP and Fragmentation

The DNS is allowed to operate over the UDP and TCP transport protocols. In general, it is preferred to use UDP where possible. UDP is a preferred due to the lower overheads of UDP, particularly in terms of the load placed on the server. However there is a limitation imposed by this protocol choice. By default, UDP can only handle responses with a DNS payload of 512 octets of less (RFC1035).

EDNS0 (RFC6891) allows a DNS requestor to inform the DNS server that it can handle UDP response sizes larger than 512 octets.

It should be noted that some clarification of the semantics of this claim is necessary. If a clients sets the value of 4096 in the EDNS0 UDP buffer size field of a query the client is telling the server that it is capable of reassembling a UDP response whose DNS payload size if up to 4,096 octets in size. It is likely that to transit such a large response the UDP response will be fragmented into a number of packets. What the client cannot control is whether all the fragments of the response will be passed through from the server to the client. Many firewalls regards IP packet fragments as a security risk, and use fragment discard rules. The client is not claiming that no such filters exist on the path form the server to the client. The EDNS0 UDP

buffer size only refers to the packet reassembly buffer size in the client's internal IP protocol stack.

Setting this EDNS0 buffer size is akin to the client saying to the server: "You can try using a UDP response up to this size, but I cannot guarantee that I will get the answer."

The requestor places the size of its UDP payload reassembly buffer (not the IP packet size, but the DNS payload reassembly size) in the query, and the server is required to respond with a UDP response where the DNS payload is no larger than the specified buffer size. If this is not possible then the server sets the "truncated" field in the response to indicate that truncation has occurred. If the truncated response includes a valid response in the initial part of the answer, the requestor may elect to use the truncated response in any case. Otherwise the intended reaction to receiving a response with the Truncated Bit set is that the client then opens a TCP session to the server and presents the same query over TCP. A client may initiate a name resolution transaction in TCP, but common client behaviour is to initiate the transaction in UDP, and use the Truncated Bit in a response to indicate that the client should switch TCP for the query.

When a UDP response is too large for the network the packet will be fragmented at the IP level. In this case the trailing fragments use the same IP level leader (including the UDP protocol number field), but specifically exclude the UDP pseudo header in the trailing fragments. UDP packet fragmentation is treated differently in IPv4 and IPv6. In IPv4 both the original sender, or any intermediate router, may fragment an IP packet (unless the Don't Fragment IP header flag is set). In IPv6 only the original sender may fragment an IP packet. If an intermediate router cannot forward a packet onto the next hop interface then the IPv6 router will generate an ICMP6 diagnostic packet with the MTU size of the next hop interface, append the leading part of the packet and pass this back to the packet's sender. When the offending packet is a UDP packet the sender does not maintain a buffer of unacknowledged data, so the IPv6 UDP sender, when receiving this message, cannot retransmit a fragmented version of the original packet. Some IPv6 implementations is to generate a host entry in the local IPv6 forwarding table, and record the received MTU in this table against the destination address recorded in the appended part of the ICMP6 message, for some locally determined cache entry lifetime. This implies that any subsequent attempts within some cache lifetime to send an IPv6 UDP packet to this destination will use this MTU value to determine how to fragment the outgoing packet. Other IPv6 implementations appear to simply discard this ICMP6 Packet Too Big message when it is received in response to an earlier UDP message.

The interaction of the DNS with intercepting middleware is sometimes quite ugly. Some firewall middleware explicitly blocks port 53 over TCP, so that any attempt by resolvers on the "inside" to use TCP for DNS queries will fail. More insidious is the case where middleware blocks the transmission of UDP fragments, so that when a large DNS response is fragmented the fragments do not reach the resolver. Perhaps one of the hardest cases for the client and server to cope with is the case where middleware blocks ICMP6 Packet Too Big messages. The sender of the large IPv6 response (TCP or UDP) is unaware that the packet was too large to reach the receiver as the ICMP6 message has been blocked, but the receiver cannot inform the sender of the problem as it never received any response at all from the sender. There is no resolution to this deadlock and ultimately both the resolver and the server need to abandon the query due to local timers.

DNS Response Size Considerations

There are a number of components in a DNS response, and the size considerations are summarized in Table 1.

8 octets	UDP pseudo header size
20 octets	IPv4 packet header
40 octets	maximum size of IPv4 options in an IPv4 IP packet header
40 octets	IPv6 packet header
512 octets	the minimum DNS payload size that must be supported by DNS
576 octets	the largest IP packet size (including headers) that must be supported by IPv4 systems
913 octets	the size of the current root priming response with DNSSEC signature
1,232 octets	the largest DNS payload size of an unfragmented IPv6 DNS UDP packet
1,280 octets	the smallest unfragmented IPv6 packet that must be supported by all IPv6 systems
1,297 octets	the largest size of a ./IN/DNSKEY response with two 2048-bit KSKs and one 1024-bit ZSK.
1,425 octets	the largest size of a ./IN/DNSKEY response with two 2048-bit KSKs and one 2048-bit ZSK.
1,452 octets	the largest DNS payload size of an unfragmented Ethernet IPv6 DNS UDP packet
1,472 octets	the largest DNS payload size of an unfragmented Ethernet IPv4 DNS UDP packet
1,500 octets	the largest IP packet size supported on IEEE 802.3 Ethernet networks

Table 1. Size components relevant to DNS

The major consideration here is to minimize the likelihood that DNS response is lost. What is desired is a situation where a UDP response with a commonly used EDNS0 buffer size can be used to form a response with a very low probability of packet fragmentation in flight for IPv4, and an equally low probability of encountering IPv6 Path MTU issues.

If the desired objective is to avoid UDP fragmentation as far as possible, then it appears that the limit in DNS payload size is 1,452 octets. Can we quantify "as far as possible"? In other words, to what extent are those resolvers that ask a DNSSEC-signed root key priming query able to accept a UDP response with a DNS payload size of 1,452 octets?

The Measurement Technique

The technique used for this measurement is one of embedding URL fetches inside online advertisements, and use the advertisement network as the distributor of the test rig (this is documented in an earlier article: http://www.potaroo.net/ispcol/2013-05/1x1xIPv6.html, and won't be repeated here). However, this approach is not easily capable of isolating the behaviour of individual resolvers who query authoritative name servers. User systems typically have two or more resolvers provided in their local environment, and these resolvers may use a number of forwarders. The user system will repeat the query based on its own local timers, and each recursive resolver also has a number of local timers. The internal forwarding structure of the DNS resolver system is also opaque, so the correlation of a user system attempting to resolve a single DNS name, and the sequence of queries that are seen at the authoritative name server are often challenging to interpret. If the user is able to fetch the Web object that is referenced in the URL then one can observe that at least one of the resolvers was able to successfully resolve the DNS name, but that does not mean that one can definitively say which resolver was the one that successfully resolved the name when the name was queried by two or more resolvers. Similarly if the web object was not fetched then it is possible that none of the resolvers were able to resolve the name, but this is not the only reason why the web fetch failed to occur. Another possibility is that the script's execution was aborted, or the user's local resolution timer expired before the DNS system returned the answer. This explains a necessary caveat that there is a certain level of uncertainty in this form of experimental measurement.

The measurement experiment was designed to measurement the capabilities of resolvers that ask questions of authoritative name servers, and in particular concentrate our attention on those resolvers

that use EDNS0 and set the DNSSEC OK flag. This has some parallels with the root server situation, but of course does not in fact touch the root servers and does not attempt to simulate a root service.

In the first experiment an authoritative server has been configured to respond to requests with a particular response size. The resolvers we are looking for are those who generate queries for an A Resource Record query with the DNSSEC OK bit set. The response to these queries is a DNS response of 1,447 octets, which is just 5 bytes under the largest DNS payload of an unfragmented Ethernet IPv6 DNS UDP packet.

Server Platform
The server is running FreeBSD 10.0-RELEASE-p6 (GENERIC)

The I/O interface supports TCP segment offloading, and is enabled with 1500 octet TCP segmentation in both IPv4 and IPv6.

The server is running a BIND server:

```
BIND 9.11.0pre-alpha <id:> built by make with '--with-aes' '--with-ecdsa'
compiled by CLANG 4.2.1 Compatible FreeBSD Clang 3.3 (tags/RELEASE_33/final 183502)
compiled with OpenSSL version: OpenSSL 1.0.1e-freebsd 11 Feb 2013
linked to OpenSSL version: OpenSSL 1.0.1e-freebsd 11 Feb 2013
compiled with libxml2 version: 2.9.1
linked to libxml2 version: 20901
```

The BIND server uses an MTU setting of 1500 octets in IPv4 and IPv6 for TCP and UDP.

The server is serving a DNSSEC-signed zone, and the specific response sizes for this experiment have been crafted using additional bogus RRSIG records against the A RR entry, on the basis that the server is unable to strip out any of these RRSIG records in its response, and will be forced to truncate the UDP response if it cannot fit in the offered buffer size.

The zone was served in both IPv4 and IPv6, and the choice of IP protocol for the query and response was left to the resolver that was querying the server.

The queries were generated using Google's online advertising network, and the advertising network was set up to deliver no less than 1 million impressions per day. Each ad impression uses a structured name string which consists of a unique left-most name part, and a common zone parent name. The use of the unique parts is to ensure that no DNS cache nor any Web proxy is in a position to serve a response from a local cache. All DNS queries for the IP addresses bound to this name are passed to the zone's authoritative name server, and similarly all http GET requests for the associated URL are also passed to the web server.

EDNS0 Buffer Size Distribution

The first distribution is a simple scan of all the offered EDNS0 buffer sizes in queries that have the DNSSEC OK flag set, for the period from the 26th March 2015 to 8th May 2015. (Figure 1). Of the 178 million queries, some 31% of queries presented an EDNS0 UDP buffer size of less than 1500 octets, and 18% of queries presented a size of 512 bytes.

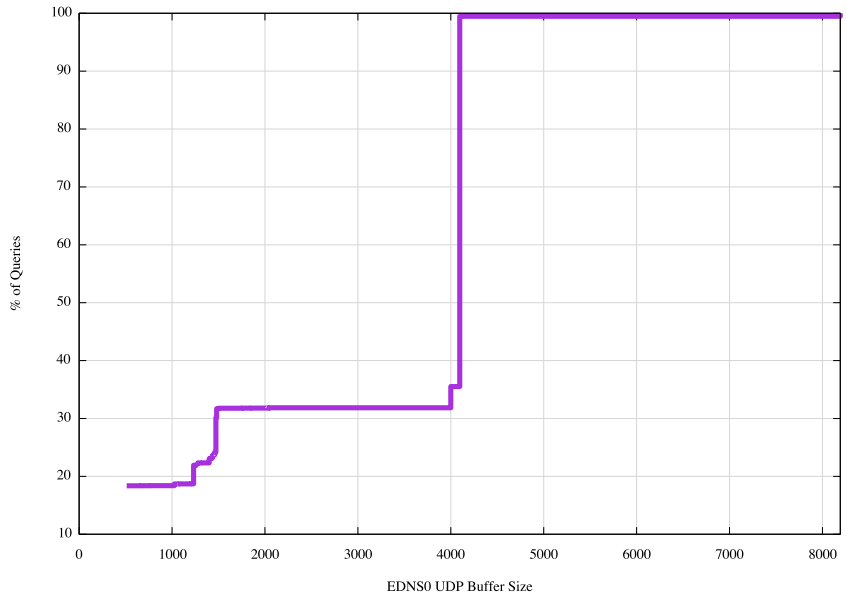

Figure 1: Cumulative distribution of EDNS0 UDP Buffer Size Distribution across all queries

This distribution reflects resolver behavior where the resolver will vary the EDNS0 buffer size and re-present the query.

A second distribution is shown in Figure 2, which shows the cumulative distribution of the maximum EDNS0 buffer sizes across the 1.5 million resolvers observed across this period. The relatively smaller proportion of small EDNS0 buffer sizes in this per-resolver distribution shows that reducing the EDNS0 buffer size in subsequent queries following a timeout is one technique a resolver can use to establish the difference between an unresponsive server and a network problem with the transmission of larger responses.

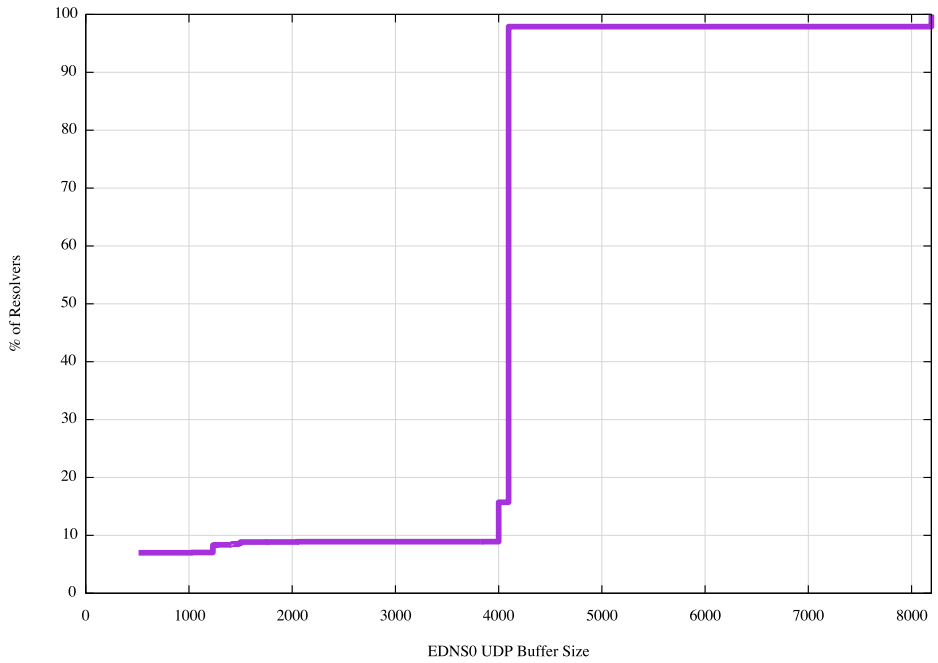

Figure 2: Cumulative distribution of EDNS0 UDP Buffer Size Distribution across all resolvers

Some 6.9% of visible resolvers already set their EDNS0 buffer size below the current size of the DNSSEC-signed root zone priming response of 913 octets

A further 1.4% of visible resolvers use a buffer size that is greater that 913 octets, and less than 1452 octets. (See Figure 3).

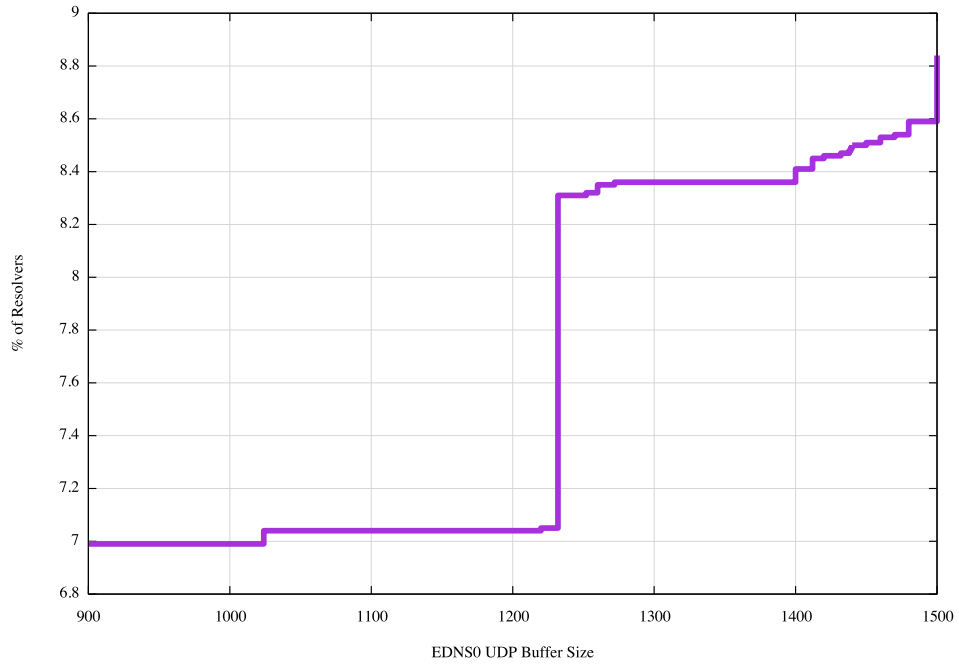

Figure 3: Cumulative distribution of EDNS0 UDP Buffer Size Distribution across all resolvers, 900 to 1500 octet range

The DNS Transport Protocol

The conventional operation of the DNS is to prefer use UDP to perform queries. Within this convention TCP is used as a fallback when the server passes a truncated response to the client in UDP. The distribution of EDNS0 UDP buffer sizes shows that 30% of all UDP queries used an EDNS0 buffer size of less than the 1,447 octet DNS response size used in the experiment, and 8.5% of all visible resolvers used an EDNS0 buffer size of less than 1,447 octets. The difference between the two numbers indicates that these resolvers who used small EDNS0 buffer sizes presented a disproportionately high number of UDP queries to the server, indicate that some resolvers evidently do not cleanly switch to TCP upon receipt of a truncated UDP response.

The experiment used an authoritative name server configured with both IPv4 and IPv6 addresses, allowing resolvers to use their local protocol preference rules to determine which protocol to use to pass queries to the name server. In this case it appears that the local preference rules appear to be weighted heavily in favor of using IPv4.

The breakdown of use of UDP and TCP, and the use of IPv4 and IPv6 over a 7 day sample period is shown in Table 2.

Number of Queries	47,826,735			
UDP: Queries	46,559,703	97%		
TCP Queries	1,267,032	3%		
			UDP	TCP
V4 Queries	47,431,919	99%	46,173,832	1,258,087
V6 Queries	394,816	1%	385,871	8,945

Table 2: DNS Transport Protocol Use

Fragmentation Signals

IPv6 IP Reassembly Time Exceeded

When middleware discards trailing fragments of an IP datagram then the destination host will receive just the initial fragment and will time out in attempting to reassemble the complete IP packet. Conventional behaviour is for hosts to send an ICMP IP Reassembly Time Exceeded message to the originator to inform the packet sender of the failure. Not all hosts reliably send this message and there is no assurance that all forms of middleware to allow this ICMP message to be passed to the sender, but this is some form of indication that middleware exists that intercepts trailing fragments of fragmented IP datagrams.

The server received no ICMP6, time exceeded in-transit (reassembly) messages across a 7 day measurement period.

IPv6 Packet Too Big

In IPv6 routers cannot fragment IPv6 packets to fit into the next hop interface. Instead, the router will place the initial part of the packet into an ICMP6 message, add the code to say that the packet is too large, and the MTU of the next hop interface. This ICMP packet is passed back to the packet's source address.

There were 205 ICMP6 packet too big messages received by the server in this experiment in this 7 day measurement period. Of these, 33 offered a 1280 MTU in the ICMP message, 1 offered 1454, 1 offered 1476, 1 offered 1500 (bug!) and the remainder offered 1480 (indicative of an IPv6 in IPv4 tunnel;).

These ICMP messages were generated by 26 IPv6 routers. The overall majority of these messages come from the two tunnel broker networks that were early providers of IPv6 via IP-in-IP tunneling (AS109 and AS 6939).

IPv4 IP Reassembly Time Exceeded

In this experiment the server received no ICMP IP Reassembly Time exceeded messages in response to responses for the A RR.

Summary

There is a relatively minor issue with IPv6 ICMP packet too big messages, where 205 messages were received out of a total of 46.5M DNS responses. These were mainly from known IPv6 tunnel broker networks, and the IPv4 query was processed without fragmentation. These messages were the outcome of a design decision to run the experiment with a 1,500 octet MTU for IPv6 UDP and TCP. It is likely that running the server with a 1280 MTU for outbound IPv6 UDP would've eliminated these IPv6 messages.

Measurement 1: DNS and Web

In this section we report on a measurement approach that combined the server's records of DNS queries and responses with the HTTP queries and responses to assemble an overall picture of the capability of resolvers who retrieve DNSSEC signatures to successfully process a response when the DNS payload approaches 1,452 octets in size (the maximum size of an unfragmented IPv6 UDP datagram). In this experiment we are using a response size of 1,447 octets.

We observe that if a DNS resolver is unable to receive a response to its query the original question will both be repeated to the original resolver to retry and the query will also be passed to alternate resolvers, assuming that multiple resolvers have been configured by the end user system or if multiple resolvers are used in the query forwarding path, or if there is some form of DNS load balancing being used in the resolver path. Ultimately, if all these resolvers are unable to resolve the name within a user-defined query timeout interval, then the user will be unable to fetch the provided URL. This observation

implies that the proportion of DNS queries where there is no matching HTTP GET operation is some approximate upper bound on the DNS resolution failure rate when large responses are used.

However, within the parameters of this test, undertaken as a background script run by a users browser upon impression of an online advertisement, there is a further factor that creates some noise that is layered upon the underlying signal. The intended outcome of the experiment is that the user's browser fetches the URL that has a DNS name where the associated DNS response is 1,447 octets, and the failure rate is determined by the absence of this fetch. However, this is not the only reason why the user's browser fails to perform the fetch. The user may switch the browser to a new page, or quit the session, or just skip the ad, all of which causes an early termination of the measurement script, which results in a failure to retrieve the web object.

We can mitigate this noise factor to some extent by observing that we are interested in DNS resolvers that perform queries with EDNS0 DNSSEC OK. The DNS name used for this measurement is DNSSEC signed, and DNSSEC-validating resolvers will perform a DS query immediately following the successful receiving of the A response. (The resolvers used by many (1 on 3) users who use resolvers that include DNSSEC OK in their A queries perform DNSSEC validation, and fetch the associated DS and DNSKEY RRs.). To identify those experiments that terminate early the experiment script also performs a further operation. The script will wait for all experiments to complete or for a 10 second timer to trigger. At this point the script will perform a final fetch to signal experiment completion.

Experiments	Timed Out	Completed	Web Seen	DS Seen	Remainder
8,760,740	329,993	8,430,747	7,615,693	815,054	0
	4%		90%	10%	0%

Table 3: DNS-to-Web Results

Across the 7 day period with just under 9 million sample points we observed no evidence that any user who used resolvers that set the DNSSEC OK bit was unable to fetch the DNS records.

There are two caveats to this result. The first is that some 4% of users were unable to complete the experiment and did not fetch a completion object, nor the initial web object, nor a DS resource record. This represents a level of uncertainty in this result. The second caveat is the nature of this experiment. This experiment measures the capability of the collection of resolvers used by each user who performed this test. The experiment does not directly measure the capability of the individual resolvers who attempted to resolve the DNS name for the user who performed the experiment.

Measurement 2: DNS Indirection

Is it possible to refine this experiment and identify individual resolvers who are unable to receive DNS responses when the response size approaches 1,452 octets?

One approach is to take a form of DNS delegation that was described in a DNS OARC presentation by Florian Maury of the French Systems Security agency ("The "indefinitely" Delegated Name Servers (IDNS) Attack," https://indico.dns-oarc.net/event/21/contribution/11/material/slides/0.pdf). The approach described there is the use a "glueless delegation", so that when a resolver is attempting to resolve a particular name it is forced to perform a secondary resolution of the name server's name, as this information is explicitly not provided in the parent zone as glue.

> For example, to resolve the name "a.b.example.com", the resolver will query the name servers for "example.com" for the name "a.b.example.com". The server will return the name servers for "b.example.com" but will be unable to return their IP addresses, as this is the glue that has been explicitly removed from the zone. Let's suppose that the name server for this zone is "nsb.z.example.com". In order to retrieve the address of the name server, the resolver will have

to query the name server for "z.example.com" for the name "nsb.z.example.com", and then use the result as the address of the server to use to query for "a.b.example.com". This is shown in Figure 4.

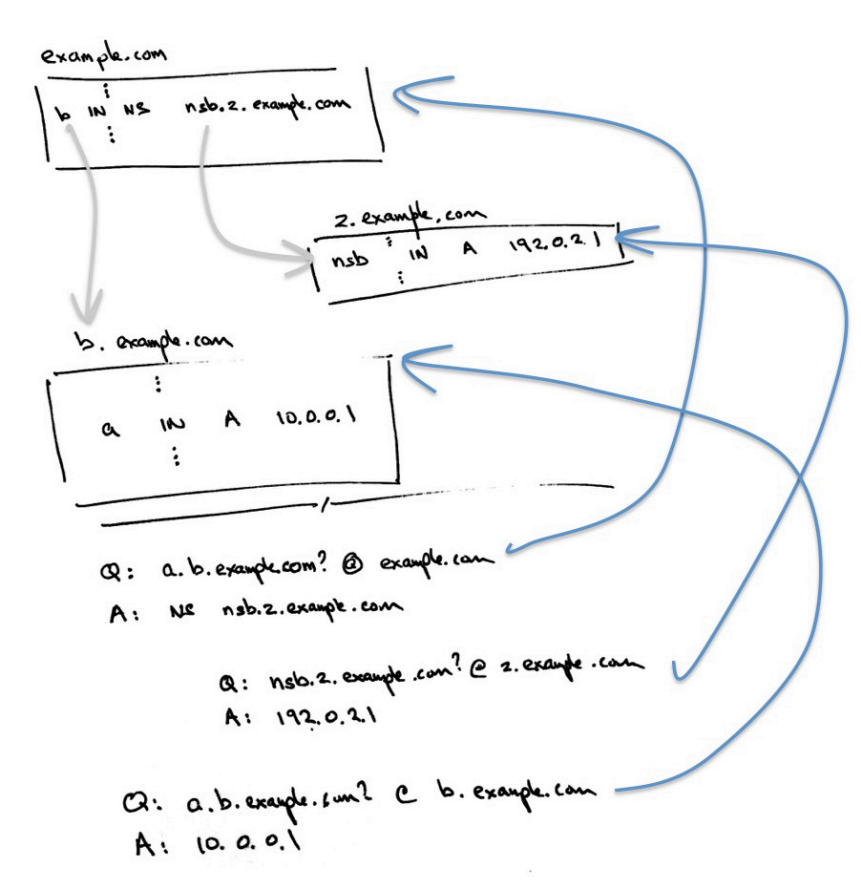

Figure 4: Example of "Glueless" Delegation

When viewed in the context of measuring the properties of DNS resolvers we note that in the above example, in order to pass a query to the name server of "b.example.com" the resolver must have been able to resolve the name "nsb.z.example.com." If we are interested in identifying those resolvers that are unable to receive a DNS response of 1,444 octets, then if we use RRSIG padding to pad the response to the "nsb.z.example.com" query, then in an ideal world the resolvers that query for "nsb.z.example.com" but do not query for "a.b.example.com" would be the resolvers that are unable to receive or process a 1,444 octet DNS response.

In this second experiment we configured two domain name families. One was a conventional, unsigned glueless delegation, where the response to the query for the name server address was a 93 octet response, which we used as the control point for the experiment. The other delegation used a DNSSEC-signed record, signed with RSA-2048, where the response to a request for the name server address was a 1,444 octet response if the resolver set the DNSSEC OK flag in the query. The names used in each experiment were structured to be unique names, so that in both cases the queries for the names, and the name server names, were not cached, and the resolution efforts were visible at the authoritative name server.

In a 109 hour period from the 21st May 2015 to the 26th May 2015 we recorded the following results:

Experiments	Resolved NS	Fail to resolve NS	DNSSEC OK Failure	Other
5,972,519	5,446,535	525,984	474,037	51,947
	91%	8%	7%	1%

Table 4: DNS Indirection Results

In this case the experiment count is the number of experiments where the resolver successfully queries for both the address of the control name server and the address of the control URL DNS name, and queried for the address of the experiment name server. In 91% of the cases the same resolver was also seen to query for the experiment URL DNS address, indicating that it had successfully received the response to the earlier name server address query. The majority of the cases that failed to receive this response were in response to name server address queries that generated a 1,444 octet response, while a smaller number of failures (slightly less than 1%) were in response to a non-DNSSEC OK query.

The nature of this particular experiment is that it is now possible to isolate the capabilities of each resolver. The statistics of resolver capability is shown in the following table

Resolvers	Resolved NS	Fail to resolve NS	DNSSEC OK Failure	Other
82,954	78,703	4,251	3,936	315
	94%	6%	5.5%	0.5%

Table 5: Resolver Statistics of DNS Indirection Results

This experiment indicates that some 6% of the resolvers that query authoritative name servers are unable to process a DNS response where the response is 1,444 octets in length. In this case this is a pool of 4,251 resolvers who can successfully follow an NS chain where the NS A response is small, but cannot follow the NS chain when they generate a query with the DNSSEC OK flag set and the response is 1,444 octets. Of this set of potentially problematical resolvers some 3,110 were only seen to perform (and fail) the experiment a single time, and it is perhaps premature to categorize these resolvers as being unable to retrieve a 1,444 octet payload on the strength of a single data point. Of the remaining 826 resolvers which exhibited this failure behavior more than once. We have the following distribution of occurrences:

Times Seen	**Count**
1	3,110
2	533
3	152
4	56
5	28
6	17
7	4
8	4
9	5
10	1
11	2
12	6
13	3
14	1
16	2
17	4
18	1
23	1
28	1
30	1
37	1
60	1
140	1
370	1

Table 6: Failing Resolver – Seen Counts

It is reasonable to surmise that if the resolver exhibits the same behavior two or more times in the context of this experiment then the possibility that this is a circumstance of experimental conditions is far lower, and the likelihood that there is a problem in attempting to pass a large DNS response to the resolver is higher. In this case some 826 resolvers, or 1% of the total count of seen resolvers falls into this category.

Of the total number of observed resolvers, some 5,237 are using IPv6, or 6% of the total. Of the 4,251 resolvers who were observed to fail to resolve the NS record, 830 resolvers, or 21% of the total number of failing resolvers used IPv6. This indicates that there are some prevalent issues with IPv6 and DNS responses that push the IPv6 packet size larger than the 1,280 octet minimum unfragmented MTU size.

These results indicate that using a 1,444 octet DNS response appears to present issues for up to 5% of all visible resolvers, who appear to be incapable of receiving the response. This is most likely for 1% of visible resolvers, and is possibly the case for the remaining 4%.

This does not necessarily imply that 5%, or even 1%, of all end users are affected. In this experiment 7,261,042 users successfully fetched the control A record, and of these some 7,172,439 successfully fetched the test record, a difference of 1%. This implies that where individual resolvers failed, the users' DNS resolution configuration passed the query to other resolvers to successfully resolve the name record.

Some 6.5% of queries were made over TCP (1,211,034 queries out of 18,620,589 queries) for the test name from those resolvers who set the DNSSEC OK flag in the query. By comparison, for the control name, 475 queries were made using TCP, from a total of 16,451,812 queries. This 6.5% figure correlates with the EDNS0 UDP buffer size distribution, where 7% of all UDP queries in this experiment had an EDNS0 buffer size of less than 1,444 octets.

Conclusions

What can we expect when the DNSSEC-signed responses to DNS queries start to approach the maximum unfragmented packet size?

This will potentially impact most users as the majority of queries (some 90% of queries in this experiment) set the EDNS0 DNSSEC OK flag, even though a far lower number of resolvers actually perform validation of the answer that they receive.

The conclusion from this experiment is that up to 5% of DNS resolvers that set the DNSSEC OK flag in their queries appear to be unable to receive a DNS response of 1,444 octets, and within this collection IPv6 is disproportionally represented. It is possible that this is due to the presence of various forms of middleware, but the precise nature of the failures cannot be established from within this experimental methodology.

However, the failing resolvers serve a very small proportion of users, the number of users who are unable to resolve a DNS name when DNS responses of this size are involved appear to be no more than 1% of all users, and likely to be significantly lower.

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 building the Internet within the Australian academic and research sector in the early 1990's. 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 and chaired a number of IETF Working Groups. He has worked as a an Internet researcher, as an ISP systems architect and a network operator at various times.

www.potaroo.net

Disclaimer